INTERNATIONAL STANDARD

ISO/IEC 24752-2

Second edition
2014-12-15

Information technology — User interfaces — Universal remote console —

Part 2: User interface socket description

Technologies de l'information — Interfaces utilisateur — Console à distance universelle —

Partie 2: Description de "socket" d'interface utilisateur

Withdrawn
STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 24752-2:2014

Reference number
ISO/IEC 24752-2:2014(E)

© ISO/IEC 2014

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 24752-2:2014
Withdrawn

COPYRIGHT PROTECTED DOCUMENT

© ISO/IEC 2014

All rights reserved. Unless otherwise specified, no part of this publication may be reproduced or utilized otherwise in any form or by any means, electronic or mechanical, including photocopying, or posting on the internet or an intranet, without prior written permission. Permission can be requested from either ISO at the address below or ISO's member body in the country of the requester.

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.org
Web www.iso.org

Published in Switzerland

Contents

Page

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 24752-2:2014
Withdrawn

© ISO/IEC 2014 – All rights reserved

Withdrawn
STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 24752-2:2014

© ISO/IEC 2014 – All rights reserved

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 24752-2:2014
Withdrawn

© ISO/IEC 2014 – All rights reserved

Withdrawn
STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 24752-2:2014

© ISO/IEC 2014 – All rights reserved

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular the different approval criteria needed for the different types of document should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives).

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation on the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the WTO principles in the Technical Barriers to Trade (TBT), see the following URL: Foreword — Supplementary information.

The committee responsible for this document is ISO/IEC JTC 1, *Information technology*, Subcommittee SC 35, *User interfaces*.

This second edition cancels and replaces the first edition (ISO 24752-2:2008), which has been technically revised.

ISO/IEC 24752 consists of the following parts, under the general title *Information technology — User interfaces — Universal remote console*:

— *Part 1: Framework*

— *Part 2: User interface socket description*

— *Part 4: Target description*

— *Part 5: Resource description*

— *Part 6: Web service integration*

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 24752-2:2014

© ISO/IEC 2014 – All rights reserved

Introduction

This is the second edition of this part of ISO/IEC 24752. The main purpose of the revision is an alignment with recent developments in the Web service area, in particular with the new ISO/IEC 24752-6 on Web service integration, along with an overall simplification of the specified technologies.

A user interface socket is an abstract concept that, when implemented, exposes the functionality and state of a target in a machine-interpretable manner. A user interface socket is independent of any specific implementation platform.

A user interface socket contains variables, commands, and notifications, optionally structured in sets that may be nested in a hierarchical fashion. The variables include all of the dynamic data a user can perceive and/or manipulate, and may also include additional dynamic supporting data that is not presented to the user. Example variables include the volume of a television, the current floor of an elevator, or an internal variable representing the current state of a transaction that is used to control dynamic features of the interface. A command is a core function that a user can request a target to perform and that cannot be represented by a variable. The commands include all target functions that can be called by users. Examples include the 'search' command of an airline reservation system or the 'seek' command of a CD player. A user interface socket does not include commands for accessing the values of the variables. There are typically no commands that simply change the values of variables. An exception would be a 'reset' operation which puts the target into a specific state. The notifications are special states where normal operation is suspended, such as an exception state. Notifications are special states triggered by the target. Examples include an announcement made by a public address system in an airport, a clock alarm, or a response to invalid input for a field of a form.

A user interface socket specification is an XML document that uses the constructs defined in this part of ISO/IEC 24752 to describe a user interface socket.

See Annex A for an example user interface socket description.

NOTE Additional information is needed before the socket can be presented to a user, including natural language labels and help text associated with the elements of the user interface. This information is provided externally to the socket description. Resources reference socket elements using the socket's name (as given in the socket descriptions 'about' attribute value, see 6.2) and the element 'id' attribute (see 7.2, 9.2 and 10.2). Refer to ISO/IEC 24752-5 for further details.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 24752-2:2014

 © ISO/IEC 2014 – All rights reserved

Information technology — User interfaces — Universal remote console —

Part 2:
User interface socket description

1 Scope

ISO/IEC 24752 is a multi-part International Standard that aims to facilitate operation of information and electronic products through remote and alternative interfaces and intelligent agents.

A user interface socket is an abstract user interface that describes the functionality and state of a device or service (target) in a machine-interpretable manner that is independent of presentation and input capabilities of a user interaction device. This part of ISO/IEC 24752 defines an Extensible Markup Language (XML)-based language for describing a user interface socket. The purpose of the user interface socket is to expose the relevant information about a target so that a user can perceive its state and operate it. This includes data presented to the user, variables that can be manipulated by the user, commands that the user can activate, and exceptions that the user is notified about. The user interface socket specification is applicable to the construction and adaptation of user interfaces.

2 Conformance

An XML file conforms to this part of ISO/IEC 24752 (i.e. is a user interface socket description) if it fulfils all of the following requirements:

a) it has an MIME type as specified in 6.1, if applicable;

b) it is coded in UCS (see 6.1);

c) its root element is the <uis:uiSocket> element (with uis representing the namespace "http://openurc.org/ns/uisocketdesc-2"), as specified in Clause 6;

d) it contains all required elements and attributes with their proper values, as specified in Clause 6;

e) if it contains recommended or optional elements or attributes with their values, these are presented as specified in Clause 6.

NOTE 1 Strict language conformance (i.e. no additional elements or attributes allowed) is not required because future versions of this part of ISO/IEC 24752 might add new elements, attributes, and values. Therefore, URC manufacturers are encouraged to implement their URCs so that unrecognized markup is ignored without failing.

NOTE 2 Target manufacturers who want to add manufacturer-specific information to a socket description beyond the elements, attributes, and values specified in this part of ISO/IEC 24752 can do so by externally providing (proprietary) resource descriptions that point into the structure of a socket description. Refer to ISO/IEC 24752-5 for details.

3 Normative references

The following documents, in whole or in part, are normatively referenced in this document and are indispensable for its application. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO 15836:2009, *Information and documentation — The Dublin Core metadata element set*

© ISO/IEC 2014 – All rights reserved

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 24752-2:2014
Withdrawn

ISO/IEC 10646:2011, *Information technology — Universal Coded Character Set (UCS)*

ISO/IEC 14977:1996, *Information technology — Syntactic metalanguage — Extended BNF*

ISO/IEC 24752-1, *Information technology — User interfaces — Universal remote console — Part 1: Framework*

ISO/IEC 24752-4, *Information technology — User interfaces — Universal remote console — Part 4: Target description*

W3C Recommendation: XML Path Language (XPath) 2.0 (Second Edition), W3C Recommendation 14 December 2010 (Link errors corrected 3 January 2011)[1)]

W3C Recommendation: XQuery 1.0 and XPath 2.0 Functions and Operators (Second Edition), W3C Recommendation 14 December 2010[2)]

W3C Recommendation: XML Schema Part 1: Structures Second Edition, W3C Recommendation 28 October 2004[3)]

W3C Recommendation: XML Schema Part 2: Datatypes Second Edition, W3C Recommendation 28 October 2004[4)]

4 Terms and definitions

For the purposes of this document, the terms and definitions given in ISO/IEC 24752-1 and ISO/IEC 24752-4, and the following apply.

4.1
context element
element to which a dependency pertains

5 Relation to other standards

5.1 Relation to XML

This specification defines an extensible Markup Language (XML) based language. Markup in XML is case sensitive.

Tag names, and attribute names and values are not localizable, i.e. they are identical for all international languages. However, the text content between tags can be language specific. As with all XML based languages, white space characters immediately surrounding tags are non-significant.

This specification makes use of the XML namespaces concept to enable the import of element and attribute names defined elsewhere.

All element and attribute names used in this document with no namespace prefix are defined by ISO/IEC 24752 series and are part of the namespace with URI reference http://openurc.org/ns/uisocketdesc-2. If not defined as the default namespace, the namespace identifier 'uis' should be used.

Throughout this document, the following namespace prefixes and corresponding namespace identifiers are used for referencing foreign namespaces:

— dc: The Dublin Core Metadata Element Set namespace (http://purl.org/dc/elements/1.1/) (Element Set defined by ISO 15836);

1) File can be accessed in http://www.w3.org/TR/2010/REC-xpath20-20101214/

2) File can be accessed in http://www.w3.org/TR/2010/REC-xpath-functions-20101214/

3) File can be accessed in http://www.w3.org/TR/2004/REC-xmlschema-1-20041028/

4) File can be accessed in http://www.w3.org/TR/2004/REC-xmlschema-2-20041028/

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 24752-2:2014
Withdrawn

© ISO/IEC 2014 – All rights reserved

— dcterms: The DCMI Metadata Terms namespace (http://purl.org/dc/terms);

— xsd: The XML Schema namespace (http://www.w3.org/2001/XMLSchema);

— xsi: The XML Schema Instance namespace (http://www.w3.org/2001/XMLSchema-instance).

For an XML Schema definition for the user interface socket description see Annex A.

5.2 XPath expressions

5.2.1 General

This specification uses XML Path Language (XPath) Version 2.0 for addressing elements within the socket. Specifically, XPath is used in describing dependencies between the elements of the socket.

XPath 2.0 syntax is used without XPath 1.0 compatibility.

5.2.2 Use of XPath 2.0 syntax and semantics

The ISO/IEC 24752 series uses the syntax and semantics of XPath 2.0, with the following additions and exceptions:

a) The XPath expressions shall be coded in UCS.

b) The static expression context (see 2.1.1 in XPath 2.0) shall be initialized with the following components:

1) "XPath 1.0 compatibility mode" shall be false.

2) The "statically known namespaces" are the namespace declarations that are in scope for he XML element that contains the XPath expression.

3) The "default element/type namespace" shall be the null namespace (which refers to types that are defined in the socket description, see Clause 11).

4) The "default function namespace" shall be the standard function namespace of XPath 2.0: http://wwww3.org/2005/xpath-functions.

5) The "in-scope schema definitions" shall only contain "in-scope schema types" with the following content: All types of namespace http://www.w3.org/2001/XMLSchema, as specified in section 2.5.1 of XPath 2.0; and the local types defined in a socket description's <xsd:schema> part (see Clause 11).

NOTE XPath 2.0 adds the following pre-defined types to the pre-defined types of XML Schema Definition Part 2: xsd:untyped, xsd:untypedAtomic, xsd:anyAtomicType, xsd:dayTimeDuration, xsd:yearMonthDuration.

6) The "in-scope variables" shall be empty.

7) The "function signatures" shall be the functions of the namespace http://www.w3.org/2005/xpath-functions, as defined in XQuery 1.0 and XPath 2.0 Functions and Operators, with exceptions as specified in 5.2.4 ; the constructor functions for all the atomic types in the "in-scope schema definitions"; and the additional functions defined in 5.2.5.

NOTE The following components of the XPath 2.0 static expression context are not used in this part of ISO/IEC 24752: "context item static type", "statically known collations", "default collation", "base URI", "statically known documents", "statically known collections", "statically known default collection type".

c) The dynamic expression context (see 2.1.2 in XPath 2.0) shall be initialized with the following components:

1) The "context item" shall be the socket set or element that the XPath expression is specified for as dependency.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 24752-2:2014
Withdrawn

© ISO/IEC 2014 – All rights reserved

2) The "variable values" shall be empty.

3) The "function implementations" shall include implementations of the functions of the namespace http://www.w3.org/2005/xpath-functions, as defined in XQuery 1.0 and XPath 2.0 Functions and Operators; the constructor functions for all the atomic types in the "in-scope schema definitions"; and the additional functions as defined in 5.2.5.

4) The "current dateTime" shall be the current time with local timezone of the URC, represented as a value of type xsd:dateTime.

5) The "implicit timezone" shall be the local timezone of the URC.

NOTE The following components of the XPath 2.0 dynamic expression context are not used in this part of ISO/IEC 24752: "context item", "context position", "context size", "Available documents", "Available collections", "Default collection".

d) There is no Data Model (XDM instance). Expressions and functions that refer to a data model instance shall not be used in socket descriptions. The context item expression (see 3.1.4 in XPath 2.0) shall not be used. Path expressions (see 3.2 in XPath 2.0) shall not be used. Node operations such as node comparison (see 3.5.3 in XPath 2.0), and the union, intersect and except operators (see 3.3.3 in XPath 2.0) shall not be used.

e) The evaluation of logical expressions (AND/OR) shall be strictly from left to right, and shall not evaluate the right operand if the result is already determined by the left operand. I.e. with an expression of the form "A and B", B shall not be evaluated if A is false; and in the case of "A or B", B shall not be evaluated if A is true. In addition, Boolean operations shall respect the "undefined" value (see 5.2.3).

f) The XPath 2.0 implementation shall be based on XML 1.0 and Namespaces in XML.

g) The XPath 2.0 implementation may support the namespace axis.

5.2.3 The undefined value

The ISO/IEC 24752 series adds the "undefined" value as a special value for all types from XPath 2.0 (see 5.2.2) and locally defined types (see Clause 11).

If any part of an XPath expression is undefined, the whole expression shall be undefined. This rule shall not apply, if, based on evaluation logic, the result of an expression is determined without evaluating any undefined part of it.

EXAMPLE The following expression will never evaluate to an undefined result. It yields true if the element with id 'myvar' is available and has the value 4, otherwise it yields false.

```
uis:hasDefinedValue('myvar') and uis:value('myvar') eq 4
```

NOTE Implementations may vary as long as the described effect is warranted. For example, an error exception could be internally raised to signal that an XPath expression yields "undefined".

5.2.4 XPath functions

The following XPath functions may be used:

— Functions of the namespace http://www.w3.org/2005/xpath-functions, as defined in XQuery 1.0 and XPath 2.0 Functions and Operators

— The constructor functions for all atomic types in the "in-scope schema definitions"

with the following exceptions:

— The function string() shall only be used with one argument.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 24752-2:2014

 © ISO/IEC 2014 – All rights reserved

— The function resolve-uri() shall not be used.

— The functions related to QName (section 11 of XQuery 1.0 and XPath 2.0 Functions and Operators), operators on NOTATION (section 13) and Functions and Operators on Nodes (section 14) shall not be used.

— The following context functions (section 13 of XQuery 1.0 and XPath 2.0 Functions and Operators) shall not be used: position, last, default-collation, static-base-uri.

The XPath 2.0 specific rules for implicit conversion between types apply.

5.2.5 Additional functions

5.2.5.1 General

The ISO/IEC 24752 series defines the following additional functions that may be used in expressing socket dependencies.

NOTE These functions are defined in the namespace "http://openurc.org/ns/uisocketdesc-2". A namespace prefix for this namespace (e.g. "uis") needs to be declared on any one of the XML elements containing the XPath expression. Note that the namespace prefix for "http://openurc.org/ns/uisocketdesc-2" must always be used for these functions since the default function namespace is "http://www.w3.org/2005/xpath-functions" (XPath 2.0 function namespace). Using the 'xmlns' attribute to declare a default XML namespace does not change the default function namespace.

5.2.5.2 xsd:anyType uis:value(string path)

This is a function which takes as its argument the path of a socket variable, command, notification, or an indexed component of it, and returns the current value of that variable, command, or notification (component).

NOTE 1 The type of the return value of uis:value() function is the same as the type of the socket element referred to by the argument 'path'. Note that, although the static return type is xsd:anyType, the dynamic type of the return value is always a more specific one (derived from the static type); it can be queried by the boolean operator 'instance of'.

The following syntax is defined for path (in Extended BNF notation, see ISO/IEC 14977):

— path = absolutePath | relativePath | shortcutPath;

— absolutePath = "/", { setPath, "/" }, elementPath;

— relativePath = ("./" | ("../", { "../" })), { setPath, "/" }, elementPath;

— shortcutPath = elementPath;

— setPath = *setId*, { "[", *setIndex* "]" };

— elementPath = normalElementPath | basicCommandPath | timedCommandPath | notifyPath;

— normalElementPath = *elementId*, { "[", *elementIndex*, "]" };

— basicCommandPath = *elementId*, { "[", *elementIndex*, "]" };

— timedCommandPath = *elementId*, { "[", *elementIndex*, "]" }, ["[ttc]"];

— notifyPath = *elementId*, { "[", *elementIndex*, "]" } ["[ttc]"];

A path shall be an absolute path, a relative path, or a shortcut path.

An absolute path shall be a slash-separated list of set ids and an element id, walking the path from the root element <uiSocket> (see 6.1) down to a socket element, with indices in square brackets wherever a set or the element has dimensions. An absolute path shall start with a slash character ("/") which stands for the root element.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 24752-2:2014
Withdrawn

A relative path shall have as context item (node) the socket element or set that the dependency is defined on. Relative paths that start with "./" have their context item as starting point for the subsequent path. Relative paths that start with "../" refer to the parent <set> of their context item as first segment in the path. Every subsequent "../" in the path refers to the next parent toward the root. No indices shall be specified for any dimensional set or element that is referred to by "./" or "../". At runtime, the missing indices (starting from the root) will be taken from the set/element owning the dependency, if not otherwise specified in the relative path. That means that relative paths occurring on dimensional sets or elements are inherently referencing elements with the same set of indices or a subset thereof (i.e. they are referring to the same "slice" of components).

A shortcut path shall omit set ids, and shall use only an element's id and indices, if any. It shall not have a leading slash character ("/"). A shortcut path shall not be used if the path includes a dimensional set.

setId is a placeholder for the 'id' attribute of a <set> element (which is an ancestor of the requested socket element). For dimensional <set> elements (i.e. <set> elements with a non-empty 'dim' attribute) *setIndex* is a placeholder for an index value of a <set> element, with the index value being compatible to the pertaining index type. <set> elements with no dimension shall have no *setIndex*.

elementId is a placeholder for the 'id' attribute of a <variable> (see 8.1), <command> (see 9.1), or <notify> element. For dimensional elements, an *elementIndex* shall be used as a placeholder for an index value of the element, with the index value being compatible to the pertaining index type. Elements with no dimension shall have no *elementIndex*.

— For <command> elements of type uis:timedCommand, the selector "[ttc]" may be added at the end of the path.

— For <notify> elements with a 'timeout' attribute, the selector "[ttc]" may be added at the end of the path.

The path argument shall be a string. The following characters shall be escaped if used as part of *setIndex* or *elementIndex,* as follows. '^' shall be used as the escape character.

— "[" shall be coded as "^["

— "]" shall be coded as "^]"

— "^" shall be coded as "^^"

NOTE 2 For hard-coded paths (i.e. the path is known at authoring time), the author can use a string literal that is enclosed in single (') or double (") quotes. For paths that are computed at runtime (e.g. when referencing variables as index values), the author can use valid XPath string operations (see 5.2) to concatenate a viable path string. See examples below.

The return value shall be the current value of the specified socket element (or its component if it is a dimensional element or has a dimensional set as ancestor). For command types that don't have state information (uis:voidCommand), an empty string shall be returned. For a command of type uis:basicCommand or uis:timedCommand and no *elementIndex*, the state (as string) of the command or its component shall be returned (see 9.3). For commands of type uis:timedCommand and an *elementIndex* of "ttc", the time to complete (as string in the xsd:duration format) of the command or its component shall be returned if it is currently defined, otherwise an empty string shall be returned. For a notification and no *elementIndex* specified, its state (as string) or the state of its component shall be returned (valid return values are "active", "inactive" and "stacked"). For a notification and an *elementIndex* of "ttc", a value indicating the remaining time to timeout (in seconds) or that of its component shall be returned.

uis:value(string path) shall evaluate to an undefined result for socket elements (or their components) that have an undefined value/state. In this case the whole expression (of which uis:value(path) is part of) shall have an undefined result.

EXAMPLE 1 A variable of type xsd:string with id "var" is nested inside two sets with ids "outerSet" and "innerSet". Neither the variable nor any of the nesting sets are dimensional. One can retrieve its value by either one of the following XPath expressions:

uis:value("/outerSet/innerSet/var")

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 24752-2:2014
Withdrawn

 © ISO/IEC 2014 – All rights reserved

uis:value(“var”)

EXAMPLE 2 A command of type uis:timedCommand with id “cmd” is nested in a set with id “setId”. One can retrieve its state by either one of the following XPath expressions:

uis:value(“/setId/cmd”)

uis:value(“cmd”)

And one can retrieve the value of its “ttc” component by either one of the following XPath expressions:

uis:value(“/setId/cmd[ttc]”)

uis:value(“cmd[ttc]”)

EXAMPLE 3 A notification with id “notifyId” is nested in a set with id “setId”. One can retrieve its state by either one the following XPath expressions:

uis:value(“/setId/notifyId”)

uis:value(“notifyId”)

EXAMPLE 4 A variable of type xsd:string with id “var” has one dimension with index type xsd:string. It is nested within a non-dimensional set element with id “setId”.

One can retrieve the value of the component with index “alpha” by either one of the following XPath expressions:

uis:value(“/setId/var[alpha]”)

uis:value(“var[alpha]”)

And the value of the component with index “3*[2^3]” by either one of the following XPath expressions (note that “[”, “^” and “]” need to be escaped):

uis:value(“/setId/var[3*^[2^^3^]]”)

uis:value(“var[3*^[2^^3^]]”)

EXAMPLE 5 Same as in example 5, but now retrieving the value of the component with an index value taken from another variable with id=“index”:

uis:value(concat(“/setId/var[“, uis:value(“index”), “]”))

uis:value(concat(“var[”, uis:value(“index”), “]”))

EXAMPLE 6 A command of type uis:timedCommand with id “cmd” has one dimension with index type xsd:integer. It is nested within a non-dimensional set element with id “setId”. One can retrieve the “ttc” field of the command with index 0 by either one of the following XPath expressions:

uis:value(“/setId/cmd[0][ttc]”)

uis:value(“cmd[0][ttc]”)

EXAMPLE 7 A variable of type xsd:string with id “var” has two dimensions with index types xsd:integer and xsd:string. It is nested within a non-dimensional set element with id “setId”. One can retrieve the value of the component with indices “3” and “none” by either one of the following XPath expressions:

uis:value(“/setId/var[3][none]”)

uis:value(“var[3][none]”)

EXAMPLE 8 A variable of type xsd:string with id “var” has two dimensions with index types xsd:integer and xsd:string. It is nested within a 1-dimensional outer set element with id “outerSet” and index type xsd:boolean, and a non-dimensional inner set element with id “innerSet”. One can retrieve the value of the component with indices “true” (for the outerSet element), “3” and “none” (for the var element) by the following XPath expression:

uis:value(“/outerSet[true]/innerSet/var[3][none]”)

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 24752-2:2014
Withdrawn

© ISO/IEC 2014 – All rights reserved

EXAMPLE 9 Same as in example 9, but now using the values of three other variables (with ids “index1”, “index2” and “index3”) as index values:

uis:value(concat(“/outerSet[”, uis:value(“index1”), “]/innerSet/var[”, uis:value(“index2”), “][”, uis:value(“index3”), “]”))

EXAMPLE 10 This example illustrates the use of relative paths. The following minimal socket defines a 1-dimensional set with an index type of xsd:integer. For every set instance, a “power” variable and a “dimmer” variable is defined. The “dimmer” value can only be changed if the power of the pertinent light is on. (The relative path “../power” in the write dependency of the dimmer variable refers to the “power” variable, with the index for the parent set being the same as for the “dimmer” variable that contains the write dependency.)

```
<uiSocket name=”http://example.com/lights/socket” id=”socket”
   xmlns=”http://openurc.org/ns/uisocketdesc-2” xmlns:xsd=”http://www.w3.org/2001/XMLSchema”>
   <set id=”lights” dim=”xsd:integer”>
      <variable id=”power” type=”xsd:boolean” />
      <variable id=”dimmer” type=”dimmerType”>
         <dependency>
            <write> value(“../power”) </write>
         </dependency>
      </variable>
   </set>
   <xsd:schema>
      <xsd:simpleType name=”dimmerType” id=”dimmerTypeId”>
            <xsd:restriction base=”xsd:integer”>
            <xsd:minInclusive value=”0” />
            <xsd:maxInclusive value=”10” />
         </xsd:restriction>
      </xsd:simpleType>
   </xsd:schema>
</uiSocket>
```

EXAMPLE 11 This example uses a relative path inside the <relevant> dependency of a set. For each set instance, one variable instance (with id=“isRelevant”) inside the set instance is controlling whether the set instance is to be presented to the user or not.

```
<uiSocket name=”http://example.com/relevantset/socket” id=”socket”
   xmlns=”http://openurc.org/ns/uisocketdesc-2” xmlns:xsd=”http://www.w3.org/2001/XMLSchema”>
   <set id=”set1” dim=”xsd:integer”>
      <variable id=”isRelevant” type=”xsd:boolean” />
      <variable id=”someData” type=”xsd:string” />
      <dependency>
         <relevant> value(‘./isRelevant”)</relevant>
   </dependency>
   </set>
</uiSocket>
```

5.2.5.3 boolean uis:hasDefinedValue(string path)

This function shall take as its argument the path of a socket variable or command and shall return “true” if the current value of that variable or of that command is defined, otherwise “false”. For command types that don’t have state information (uis:voidCommand), “false” shall be returned.

The argument path denotes an XPath expression that shall evaluate to a string.

NOTE Since XPath defines the ‘or’ and ‘and’ operators so that the right operand is not evaluated if the left operand pre-determines the result, one can check for undefined values/states by calling uis:hasDefinedValue(id) before calling uis:value(id).

EXAMPLE The following expression will never evaluate to an undefined value. It yields true if the command with id ‘reset’ has an undefined state or the state ‘succeeded’.

not(uis:hasDefinedValue(‘reset’)) or uis:value(‘reset’) eq ’succeeded’

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 24752-2:2014
Withdrawn

 © ISO/IEC 2014 – All rights reserved

5.2.5.4 boolean uis:isAvailable(string path)

This function shall take as its argument the path of a socket element (variable, command or notification) and shall return “true” if the socket element is available at runtime, otherwise “false”.

The argument path denotes an XPath expression that shall evaluate to a string.

NOTE 1 Socket elements can be marked as optional in the socket description. For those socket elements, it can be determined at runtime by calling uis:isAvailable() whether they are available or not.

NOTE 2 Since XPath defines the ‘or’ and ‘and’ operators so that the right operand is not evaluated if the left operand pre-determines the result, one can check for the availability of a socket element by calling uis:isAvailable(id) before calling uis:hasDefinedValue() or uis:value(id).

EXAMPLE The following expression will only evaluate to true, if the variable ‘powerstate’ is available, is not undefined and has the value “standby”.

uis:isAvailable(‘powerstate’) and uis:hasDefinedValue(‘powerstate’) and uis:value(‘reset’) eq ’standby’

5.2.5.5 boolean uis:isNotifyActive()

This function shall take no argument, and shall return a Boolean. It shall return “true” if any <notify> element in the socket is active, and “false” if no <notify> element is active.

NOTE The XPath expression “not(uis:isNotifyActive())” can be useful to check for normal mode (i.e. no notification is active).

5.2.5.6 boolean uis:sessionForward(string type, string uri)

This function states session forwarding (see ISO/IEC 24752-1). The value of the type argument shall be either “destructive” or “spawn”. The value of the uri argument shall be the name (URI) of the socket that the URC is forwarded to. The return value shall be “true” if the target has sent the specified session forward event to the URC, otherwise “false”. This function may be used in postconditions of socket commands (see 9.9.5), to provide a hint to the URC that the command could trigger a session forwarding.

6 Structure of a socket description

6.1 General

A socket description shall be an XML document with the <uiSocket> element as its root element of namespace “http://openurc.org/ns/uisocketdesc-2”.

A socket description document shall be coded in UCS according to ISO/IEC 10646. For character encoding, “UTF-8” or “UTF-16” shall be used.

A socket description shall have a MIME type of “application/urc-uisocketdesc+xml”, if applicable. The ‘charset’ parameter (see IETF RFC 3023) should be used to specify the character encoding of the socket description. Its value shall be “utf-8” or “utf-16”. If the ‘charset’ parameter is absent, the procedure specified in “Extensible Markup Language (XML) 1.0 (Fifth Edition)”, section 4.3.3 shall be followed to determine the character encoding.

The following sections describe the attributes and sub elements of the <uiSocket> element in more detail.

EXAMPLE See the following:

```
<uiSocket
   about=”http://example.com/thermometer/socket”
   id=”socket”
   xmlns=”http://openurc.org/ns/uisocketdesc-2”
   xmlns:xsd=”http://www.w3.org/2001/XMLSchema”
   xmlns:dcterms=”http://purl.org/dc/terms/”
```

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 24752-2:2014
Withdrawn

```
   <dcterms:conformsTo>http://openurc.org/ns/
uisocketdesc-2/isoiec24752-2-2013/dcterms:conformsTo
   <!- … variables, commands, and notify elements, optionally contained in set elements … ->
   <!- … internal xsd type definitions … ->
</uiSocket>
```

6.2 The ‘about’ attribute

The <uiSocket> element shall have an ‘about’ attribute. The ‘about’ attribute references the socket that is described by the socket description. The value of the ‘about’ attribute shall be specified as element content and shall be the name of the socket expressed as a URI. The URI may or may not be resolvable.

NOTE 1 Target manufacturers are encouraged to make the socket descriptions of their products publicly available by posting the socket description at the socket’s name URI.

Typically this URI is derived from the target’s URI by concatenation.

EXAMPLE http://example.com/target/socket; if the target’s URI is http://example.com/target.

NOTE 2 The same URI is used as a socket reference in the socket description (‘about’ attribute of <uiSocket> element), and the name of the socket in the target description (‘name’ attribute of the corresponding <socket> element). Refer to part 4 of this ISO/IEC 24752 for details on the target description.

6.3 The ‘id’ attribute

The <uiSocket> element shall have an ‘id’ attribute that specifies an identifier that is unique among all ids in the socket description.

NOTE The identifier is used to specify external resources for the socket description root element, as exemplified in part 5 of ISO/IEC 24752.

6.4 The ‘sufficient’ attribute

The <uiSocket> element may have a ‘sufficient’ attribute of type Boolean. Its default value shall be “false”.

The value “true” indicates that all commands in the socket description that have no ‘sufficient’ attribute themselves, are sufficiently specified as described in 9.5.

NOTE The ‘sufficient’ attribute for <uiSocket> is provided for the convenience of the author. It can be overwritten by separate ‘sufficient’ attributes on the individual <command> elements (see 9.1).

6.5 The ‘complete’ attribute

The <uiSocket> element may have a ‘complete’ attribute of type Boolean. Its default value shall be “false”.

The value “true” indicates that all commands in the socket description that have no ‘complete’ attribute themselves, are completely specified as described in 9.6.

NOTE The ‘complete’ attribute for <uiSocket> is provided for the convenience of the author. It can be overwritten by separate ‘complete’ attributes on the individual <command> elements (see 9.1).

6.6 The ‘extends’ attribute

A socket may extend one or multiple other sockets (i.e. inherit from one or multiple sockets). The socket that is an extension of one or multiple other sockets is called “sub-socket”, and the sockets that it extends are called “super-sockets”.

A sub-socket shall inherit the following items from its super-sockets:

— All sets

— All variables, commands, and notifications (with their types)

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 24752-2:2014
Withdrawn

 © ISO/IEC 2014 – All rights reserved

— All internal type and element definitions (from the <xsd:schema> part)

NOTE 1 The socket container (i.e. <uiSocket> element) is not inherited.

NOTE 2 Namespace references are not inherited. If a sub-socket adds a variable with an external type definition it must declare the namespace for the external type even if that namespace has already been declared in a super-socket.

The <uiSocket> element may have an 'extends' attribute with a space-separated list of socket names (URIs), specifying an ordered list of super-sockets that the described sub-socket inherits from.

EXAMPLE The socket with name "http://example.com/thermometerplus/socket" extends the socket with name "http://example.com/thermometer/socket". Ellipses indicate code omissions.

```
<uiSocket
   about="http://example.com/thermometerplus/socket"
   extends="http://example.com/thermometer/socket"
   id="socket"
   xmlns="http://openurc.org/ns/uisocketdesc-2">
   …
</uiSocket>
```

In the case that the same identifier (value of 'id' attribute) occurs on socket elements of one or multiple super-sockets and/or the sub-socket, the last occurrence shall overwrite previous occurrences, i.e. the previous occurrences of socket elements are not inherited by the sub-socket. Super-sockets come first in their order, then the sub-socket. The order between the super-sockets is defined by the order in which the names (URIs) of the sockets occur in the value of the 'extends' attribute.

If a sub-socket element overwrites a super-socket element (by using the same identifier), the respective super-socket element shall not be inherited into the sub-socket, i.e. the resulting sub-socket contains only the respective element as specified in the sub-socket description.

If an element of a super-socket overwrites one or multiple elements of other super-sockets by using the same identifier, the overwritten elements (i.e. all but the last) shall be deemed as not existing, and only the respective element of the last super-socket shall be inherited into the sub-socket (unless there is an element with the same identifier defined in the sub-socket in which case the sub-socket element overwrites all super-socket elements with the same identifier).

6.7 The <dcterms:conformsTo> element

The <uiSocket> element shall have a subelement <dcterms:conformsTo> that specifies a reference to an established standard to which the socket conforms. The value, a URI, is provided as element content. The value "http://openurc.org/ns/uisocketdesc-2/isoiec24752-2-2013" indicates that the described socket conforms to this part of ISO/IEC 24752.

NOTE 1 The value of the <dcterms:conformsTo> element can be used when testing for conformance of a socket description.

NOTE 2 <dcterms:conformsTo> conforms to the Dublin Core metadata element refinement conformsTo, http://purl.org/dc/terms/conformsTo which is a refinement of the Dublin Core element http://purl.org/dc/elements/1.1/relation

EXAMPLE <dcterms:conformsTo> http://openurc.org/ns/uisocketdesc-2/isoiec24752-2-2013/dcterms:conformsTo

6.8 The <dcterms:modified> element

The <uiSocket> element may include a <dcterms:modified> element. This indicates that the socket description document has been modified from its original version, while still using the same reference socket URI in its 'about' attribute (see 6.2). Its content is of type xsd:date or xsd:dateTime.

NOTE <dcterms:modified> conforms to the Dublin Core metadata element refinement modified, http://purl.org/dc/terms/modified.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 24752-2:2014

© ISO/IEC 2014 – All rights reserved

EXAMPLE `<dcterms:modified> 2004-01-30 </dcterms:modified>`

A socket description should remain stable wherever possible. A socket description document that is changed shall be assigned a new about URI (as specified in 6.2) or shall change the value of the <dcterms:modified> element.

6.9 Socket description properties from DCMI

Any element and element refinement (except <dcterms:conformsTo> and <dcterms:modified> which are referenced above) from ISO 15836, Dublin Core Metadata Element Set, or the set of Dublin Core Metadata Initiative (DCMI) Metadata Terms may be used to describe a socket description, if appropriate. Each of them may occur multiple times as subelements of the <uiSocket> element.

In particular, the following Dublin Core Metadata Elements may be applied:

— <dc:creator> — specifying the author of the socket description

— <dc:publisher> — specifying the provider of the socket description

— <dc:contributor> — specifying a co-author of the socket description

— <dcterms:hasVersion> — specifying the name of an alternate socket for accessing the same or similar functionality

— <dcterms:isVersionOf> — specifying the name of the primary socket among a set of alternate sockets for accessing the same or similar functionality

6.10 <variable> , <command> , <notify>, and <set> elements

The <uiSocket> element shall contain a complete set of <variable> (see Clause 8), <command> (see Clause 9) and <notify> (see Clause 10) elements capturing all of the information that can be perceived or manipulated by a user, and all of the commands available to a user of the target. These elements may be structured via the <set> element (see Clause 7) which may be nested. An implicit ordering shall be implied by the order in which the elements appear within the socket description document.

6.11 XSD type schema elements

A socket description may contain an XSD type schema declaration. This is used to define internal types within the socket as described in Clause 11.

6.12 Platform-specific mapping information for sockets

The <mapping> element may be used any number of times as subelement of <uiSocket> to include platform-specific mapping information pertaining to the socket.

A <mapping> element shall have a 'platform' attribute whose value is not restricted by the ISO/IEC 24752 series.

A <mapping> element may have arbitrary element content and subelements. However, subelements shall be from namespaces other than the uis namespace.

NOTE 1 Socket descriptions that contain platform specific mapping information lose their platform neutrality. Although multiple mappings may be specified in a socket description (one for each platform) it is recommended to consider other mechanisms of specifying the binding to platform-specific technologies. For example, mapping information may be provided in an external file with references to the elements of the socket description.

NOTE 2 Vendors and platform carriers are strongly discouraged from using the <mapping> element for embedding active or executable content in a socket description. This would introduce a security risk for components parsing such a socket description, and executing such content.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 24752-2:2014
Withdrawn

 © ISO/IEC 2014 – All rights reserved

7 Sets

7.1 General

Sets express an internal hierarchical structure of a user interface socket. Sets may be nested (i.e. a <set> element may appear within another <set> element).

NOTE The <set> element defined in the ISO/IEC 24752 series is intended to provide a hierarchical structure internal to a socket. "Grouping resources" (as defined in ISO/IEC 24752-5) can be used for specifying presentational grouping structures (which may be non-hierarchical), or presentational grouping can be contained in a user interface implementation description (UIID, see ISO/IEC 24752-1).

7.2 Attribute 'id'

A <set> element shall have an 'id' attribute.

The 'id' attribute shall be of type ID as defined by XML Schema Part 2: Datatypes. It provides an identifier which is used to refer to the element within the socket description and as a means of binding externally defined resources such as labels. The 'id' value shall be unique among all 'id' and 'name' attributes within the socket description.

NOTE 'id' attribute values are not normally presented to users and need not be human comprehensible.

EXAMPLE <set id="mainSet">

7.3 Attribute 'dim'

The 'dim' attribute specifies a set as dimensional (with one or more dimensions). At runtime, the set is instantiated once for every possible combination of its index values. This results in multiple values for every socket element that is (directly or indirectly) contained in the dimensional set.

NOTE 1 There is no explicit ordering defined for the instances resulting from a dimensional set. However, ordering them based on the order of the index values is recommended for the target — in conveying the instances to the URC — and for the URC — in presenting the instances to the user (see ISO/IEC 24752-1).

NOTE 2 The <insert> dependency specifies whether the URC can change the set of indices at runtime (see 7.4.4).

If dimensional sets are nested, the values for the socket elements multiply based on every existing combination of the set of index values of all dimensional sets involved, plus the index values of the socket element itself, if it is dimensional.

NOTE 3 In other words, the largest possible set of values for a particular socket element results from the product of all index types that occur when walking the path from the <uiSocket> element down to the particular socket element (see definition of path in 5.2.5.2). However, not every possible combination may actually occur at runtime.

NOTE 4 A dimensional set is also called a "repeating set".

The 'dim' attribute may be present for <set> elements. If present, it shall contain a non-empty, ordered, space-separated list of type references that are the set's index types. The first reference specifies the index type for the first dimension, the second type for the second dimension, and so on. Valid index type references are: (a) the name of a type that is defined in the <xsd:schema> part of the socket description, or (b) the fully qualified name (QName) of an external type.

NOTE 5 A set may have an index of a type with an infinite set of possible index values. However, at runtime a finite subset of the index type values will be used as actual indices of the dimensional set.

EXAMPLE A sound mixer application mixes two input streams into one output stream. Each stream has 2 audio settings (volume and loudness) and a "break" command that mutes the channel for 5 s.

```
<set id="stream" dim="streamType">
   <variable id="volume" type="volumeType" />
   <variable id="loudness" type="xsd:boolean" />
```

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 24752-2:2014
Withdrawn

© ISO/IEC 2014 – All rights reserved

```
   <command id="break" type="uis:timedCommand" />
 </set>
 …
<xsd:schema>
   <xsd:simpleType name="streamType">
      <xsd:restriction base="xsd:string">
      <xsd:enumeration value="Input1" />
      <xsd:enumeration value="Input2" />
      <xsd:enumeration value="Output" />
      </xsd:restriction>
   </xsd:simpleType>
   <xsd:simpleType name="volumeType">
      <xsd:restriction base="xsd:integer">
      <xsd:minInclusive value="0" />
      <xsd:maxInclusive value="100" />
      </xsd:restriction>
   </xsd:simpleType>
</xsd:schema>
```

NOTE 6 References (e.g. from atomic resources) to dimensional sets refer to the individual components of the set. In the above example, the URI "http://example.com/mixer/socket#stream" would refer to any individual set component composed of the 'volume' variable, the 'loudness' variable and the 'break' command. However, the URI "http://example.com/mixer/socket#dims(stream)" would refer to the overall set with all components, i.e. to all channels as a compound entity. See ISO/IEC 24752-5 for details.

NOTE 7 The order, in which the components of a dimensional set are presented to the user, is based on the defined ordering of the pertinent index type(s). See ISO/IEC 24752-1 for details.

7.4 Set dependencies

7.4.1 General

A <dependency> element may be present within a <set> element. If present, it shall occur exactly once.

7.4.2 The <relevant> dependency

A <relevant> dependency may be present as subelement of a <dependency> element on sets. If present, it shall occur exactly once.

The content of the <relevant> element shall be as specified for the <relevant> dependency of variables (see 8.9.2).

A set shall inherit its <relevant> dependency to its members (applies to sets, variables, commands and notifications), unless the member specifies a <relevant> dependency on its own.

NOTE Inheritance of dependencies provides a convenient way for authors to specify a dependency once (on the set level) that is common to all (or most) of its members.

7.4.3 The <write> dependency

A <write> dependency may be present as subelement of a <dependency> element on sets. If present, it shall occur exactly once.

The content of the <write> element shall be as specified for the <write> dependency of variables (see 8.9.2).

A set shall inherit its <write> dependency to its members (applies to sets, variables and commands), unless the member specifies a <write> dependency on its own.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 24752-2:2014
Withdrawn

 © ISO/IEC 2014 – All rights reserved

7.4.4 The <insert> dependency

The <insert> dependency specifies whether the user can modify the pertaining set of valid index combinations at runtime (with the pertaining set being the indices specified by the 'dim' attribute on the corresponding <set> element).

The content of the <insert> dependency shall be an XPath expression that evaluates to a Boolean value which may change during a session. If it evaluates to false, then it is inappropriate to add or delete an index combination; otherwise it is appropriate to attempt to add or delete an index combination, although there is no guarantee that it will succeed.

NOTE 1 "Adding an index combination" means to add a component of every socket element contained in the corresponding <set> element, for a particular new index combination within the index space defined by the corresponding 'dim' attribute. Note that the initial values of the new components are determined by the target. "Removing an index combination" means to delete a component of every socket element contained in the corresponding <set> element, for a particular index combination that is currently occurring.

An <insert> dependency may be present as subelement of a <dependency> element on dimensional sets (i.e. sets with 'dim' attribute, see 7.3). If present, it shall occur exactly once.

Its default value shall be "false()".

NOTE 2 The <insert> dependency on a set specifies if an index can be inserted (or removed) for those dimensions only that pertain to the set. If a socket element underneath this set has other dimensions (either from other sets that are ancestors of that element, or because it is a dimensional element itself), these can have conflicting <insert> dependencies. In this case, it is not clear whether adding or removing an index combination for that element is appropriate, and it is up to the target to decide on a case-by-case basis.

7.5 Platform-specific mapping information for sets

The <mapping> element may be used any number of times as subelement of <set> to include platform-specific mapping information for the set.

A <mapping> element shall have a 'platform' attribute whose value is not restricted by ISO/IEC 24752.

A <mapping> element may have arbitrary element content and subelements. However, subelements shall be from namespaces other than the uis namespace.

NOTE 1 Socket descriptions that contain platform specific mapping information lose their platform neutrality. Although multiple mappings may be specified in a socket description (one for each platform) it is recommended to consider other mechanisms of specifying the binding to platform-specific technologies. For example, mapping information may be provided in an external file with references to the sets and elements of the socket description.

NOTE 2 Vendors and platform carriers are strongly discouraged from using the <mapping> element for embedding active or executable content in a socket description. This would introduce a security risk for components parsing such a socket description, and executing such content.

7.6 Set properties from DCMI

Any element and element refinement from ISO 15836, Dublin Core Metadata Element Set, or the set of Dublin Core Metadata Initiative (DCMI) Metadata Terms may be used to describe a <set> element, if appropriate. Each of them may occur multiple times as subelement of the <set> element.

7.7 Set members

A <set> element shall contain a non-empty set of <set> , <variable> (see 8.1), <command> (see 9.1) or <notify> elements as members.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 24752-2:2014
Withdrawn

© ISO/IEC 2014 – All rights reserved

8 Variables

8.1 General

Variables are used to expose the state of a target to a URC. The value of a variable can be displayed to the user of the URC, and the user can change the target's state by changing the variable's value.

The <variable> element may occur as subelement of <set> (see Clause 7), or as subelement of <uiSocket> (see 6.1).

EXAMPLE Examples of variables include the current channel showing on a television set, and a value (not shown to the user) indicating whether the current user has accepted the terms of a license.

8.2 The 'id' attribute

A <variable> element (see 8.1) shall have an 'id' attribute.

The 'id' attribute shall be of type ID as defined by XML Schema Part 2: Datatypes. It provides an identifier which is used to refer to the element within the socket description and as a means of binding externally defined resources such as labels. The 'id' value shall be unique among all 'id' and 'name' attributes within the socket description.

NOTE 'id' attribute values are not normally presented to users and need not be human comprehensible.

EXAMPLE <variable id="tvVolume" type="xsd:decimal"/>

8.3 The 'type' attribute

8.3.1 General

A <variable> element (see 8.1) shall have a 'type' attribute.

If the variable has a 'dim' attribute (see 8.8) the 'type' attribute specifies the type of any one of the variable's values ("homogenous array").

8.3.2 Simple types

Any simple type defined in XML Schema Part 2 may be used as variable type, or any other simple type derived from one of these types (including derivation by list or union).

NOTE The usage of datatype declarations borrowed from XML Schema Part 2 does not imply that the values are coded as defined by XML Schema. This is implementation dependent and not defined by ISO/IEC 24752 series. For example, a variable of type xsd:integer may be represented in the socket as a binary value, and not as a string.

8.3.3 Space-Separated Value Lists

Any type that has been derived by list according to XML Schema Part 2 may be used as variable type. If used for a variable, a list type shall be defined in the type definition part of the socket description (see Clause 11). Alternatively, the pre-defined type uis:stringList may be used if the set of string items is not restricted (see 11.3).

For the purpose of expressing dependencies in XPath 2.0 syntax (see 5.2), the value of a list variable shall be treated as a sequence of the atomic type from which it is derived (see 5.2.2).

8.3.4 Comma-Separated Value (CSV) Lists

The pre-defined type uis:csvlist may be used as variable type referring to a list of string values, represented as concatenation of individual strings separated by single comma (',') characters.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 24752-2:2014
Withdrawn

 © ISO/IEC 2014 – All rights reserved

Escaping conventions use the backslash character, '\' (UTF-8 character code 0x5C), as follows: a backslash character ('\') is represented as '\\' and a comma (',') as '\,' in individual string entries in CSV lists. Any occurring white space characters are construed as part of the string entries.

The empty string ("") shall be interpreted as empty list (no entries).

For the purpose of expressing dependencies in XPath 2.0 syntax (see 5.2), the value of a uis:csvlist variable shall be treated as of type xsd:string (see 5.2.2).

NOTE 1 The XPath 2.0 provided functions for string manipulations can be used to inspect values of comma-separated list variables, when used in dependency expressions. URC implementations may make the individual values of the comma-separated list available to UIIDs through its API, but this is implementation-specific.

NOTE 2 Type uis:csvlist is an alternative to uis:stringList (see 11.3).

NOTE 3 uis:csvlist is especially useful for UPnP AV environments where some values are conveyed as comma-separated value lists. Note that uis:csvlist, however, does allow for entries of type string only.

NOTE 4 In contrary to ISO/IEC 24752 series, the UPnP AV specifications leave it open whether the empty string should be interpreted as an empty list or a list with one empty-string entry.

EXAMPLE 1 "1,2,3" represents a list with 3 string entries: "1", "2" and "3".

EXAMPLE 2 "Smith\, Fred,Jones\, Davey" represents a list of two strings: "Smith, Fred" and "Jones, Davey".

EXAMPLE 3 "alpha, beta" represents a list of 2 strings, each with two leading spaces: "alpha" and "beta".

EXAMPLE 4 ",," represents a list of 3 empty string entries (each "").

EXAMPLE 5 "" represents an empty list (no entries).

8.3.5 Stream types

For socket variables that convey streams of text, audio or video, the following predefined types may be used as values for the 'type' attribute:

— "uis:textStreamOut": Text stream flowing from the target to the URC (client).

— "uis:textStreamIn": Text stream flowing from the URC to the target.

— "uis:audioStreamOut": Audio stream flowing from the target to the URC.

— "uis:audioStreamIn": Audio stream flowing from the URC to the target.

— "uis:videoStreamOut": Video stream flowing from the target to the URC.

— "uis:videoStreamIn": Video stream flowing from the URC to the target.

— "uis:multiMediaStreamOut": Stream that holds a synchronized combination of video, audio and text flowing from the target to the URC.

— "uis:multiMediaStreamIn": Stream that holds a synchronized combination of video, audio and text flowing from the URC to the target.

The format and transmission protocol of any of these stream types is implementation specific, and may not be known before runtime. However, if the set of target-supported formats is known before runtime, the <dc:format> element from DCMI (see 8.12) should be used (any appropriate number of times) as subelement of <variable> (see 8.1) to specify the available formats, each coded as MIME type (see IETF RFC 2046).

Withdrawn
STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 24752-2:2014

© ISO/IEC 2014 – All rights reserved

Also, if the stream is known to be specific to a natural language before runtime, the <dc:language> element from DCMI (see 8.12) should be used as subelement of <variable> (see 8.1) to specify the language.

NOTE 1 It is recommended that implementations of sockets and TUNs allow for streaming format negotiation between a URC and a target at runtime.

NOTE 2 The presence of a stream typed variable in the socket description is for descriptive purposes and does not imply that the pertaining stream "runs through" a socket implementation at runtime. Implementations are free to handle streams in a way that is suitable for their specific requirements. For example, a socket implementation may help to set up a connection for streaming, with the actual stream bypassing the socket implementation for performance reasons.

8.3.6 Socket-internal types

Oftentimes the XSD-provided types (see 8.3.2) are not sufficient to express a variable's value space in terms of allowed and disallowed values. In these cases a variable's 'type' attribute may reference socket-internal types, i.e. types that are defined in a separate section of the socket description, as specified in 10.12. The reference to a socket-internal type shall consist of the type's name, without any namespace prefix.

8.3.7 Imported types

Types defined in other namespaces may be referenced by their full qualified name (QName). In this case the namespace URI shall be fully specified, and the location of the pertinent XML schema definition (XSD) file should be given via the 'xsi:schemaLocation' attribute.

NOTE The socket can treat values of imported types internally as of type string, in particular when used inside XPath expressions (see 5.2) for dependencies. No checking against well-formedness or validity should be assumed. However, it can make values of imported types available in a special format through an API. For example, it can offer a DOM-based API for variables of imported types, but this is implementation-specific.

EXAMPLE This variable contains a description of the content of a content directory service. The description is of type DIDLType, as defined by the DIDL ("Digital Item Declaration Language") namespace urn:mpeg:mpeg21:2002:02-DIDL-NS. The pertinent XSD file is available at http://standards.iso.org/ittf/PubliclyAvailableStandards/MPEG-21_schema_files/did/didl.xsd.

```
<uiSocket
   about="http://example.com/cds"
   id="socket"
   xmlns="http://openurc.org/ns/uisocketdesc-2"
   xmlns:didl="urn:mpeg:mpeg21:2002:02-DIDL-NS"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="urn:mpeg:mpeg21:2002:02-DIDL-NS
      http://standards.iso.org/ittf/
PubliclyAvailableStandards/MPEG-21_schema_files/did/didl.xsd">
      <set id="firstSet">…
      <variable id="content" type="didl:DIDLType" />
   </set>
   …
</uiSocket>
```

8.4 The 'secret' attribute

<variable> (see 8.1) elements may have a 'secret' attribute. It is a Boolean indicating whether a variable's value is sensitive regarding security and privacy, and requires protection to support the user's privacy.

The default value shall be "false".

EXAMPLE A password could be declared as:

```
   <variable id="accountPassword" type="xsd:string" secret="true" />
```

If the variable has a 'dim' attribute (see 8.8) the 'secret' attribute applies to any one of the variable's values.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 24752-2:2014
Withdrawn

 © ISO/IEC 2014 – All rights reserved

8.5 The ‘sensitive’ attribute

A <variable> element (see 8.1) may have a ‘sensitive’ attribute. It is a Boolean indicating whether the variable shall be presented to the user under all circumstances.

The default value is “false”.

EXAMPLE A variable marked with sensitive=”true” could represent safety or warranty information that is to be presented to the user:

```
<variable id=”warrantyPeriod” type=”xsd:duration” sensitive=”true” />
```

If the variable has a ‘dim’ attribute (see 8.8) the ‘sensitive’ attribute applies to any one of the variable’s values.

8.6 The ‘optional’ attribute

A variable may have an ‘optional’ attribute of type Boolean. Its default value shall be “false”.

If optional=“true”, the variable may not be available at runtime due to various constraints. If it is not available at runtime, its value shall be undefined and the unavailability shall be indicated to the URC at runtime.

EXAMPLE If used in dependency expressions, an optional variable can be checked if it has a defined value before accessing the value itself:

```
<write> uis:hasDefinedValue(‘myvar’) and uis:value(‘myvar’) eq 4 </write>
```

NOTE 1 Examples for constraints that impact the availability of a socket element are: access control, different products using a common socket description but with some implementation variation (such as in UPnP DCPs).

NOTE 2 The availability of a socket element will be indicated at runtime to the URC through TUN-specific means (see ISO/IEC 24752-1).

If the variable has a ‘dim’ attribute (see 8.8) the ‘optional’ attribute applies to the variable as a whole, i.e. either it exists as dimensional variable or not at all.

8.7 The ‘final’ attribute

Some variables are provided initially by the target at session opening, and are never changed during a session. These should be marked with final=”true”.

A <variable> element (see 8.1) may have the Boolean ‘final’ attribute. Its default value shall be “false”.

EXAMPLE `<variable id=”serialNumber” type=”xsd:integer” final=”true” />`

NOTE A final variable is not writeable by the user and hence has no <write> dependency (see 8.9.3).

If the variable has a ‘dim’ attribute (see 8.8) the ‘final’ attribute applies to the variable as a whole, i.e. all of its values are initialized by the target at session opening and will never change during a session.

8.8 The ‘dim’ attribute

The ‘dim’ attribute specifies variables as dimensional (with one or more dimensions). At runtime a dimensional variable has multiple values, each one for a particular combination of indices.

NOTE 1 A 1-dimensional variable is an array with one index, and a 2-dimensional variable is a table (though the table may be sparsely populated).

NOTE 2 There is no explicit ordering defined for the values resulting from a dimensional variable. However, ordering the values based on the order of the index values is recommended for the target — in conveying the values to the URC — and for the URC — in presenting the values to the user (see ISO/IEC 24752–1).

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 24752-2:2014
Withdrawn

© ISO/IEC 2014 – All rights reserved

NOTE 3 The largest possible set of values for a particular socket variable results from the product of all index types that occur when walking the path from the <uiSocket> element down to the particular socket variable (see definition of path in 5.2.5.2). However, not every possible combination may actually occur at runtime.

The 'dim' attribute may be present for variables. If present, it shall contain a non-empty, ordered, space-separated list of type references that are the variable's index types. The first reference specifies the index type for the first dimension, the second type for the second dimension, and so on. Valid index type references are: (a) the name of a type that is defined in the <xsd:schema> part of the socket description, and (b) the fully qualified name (QName) of an external type.

NOTE 4 The index type may specify an infinite number of possible index values (e.g. xsd:integer has infinite values). However, the actual index values at runtime are a finite subset of the values allowed by the index type.

NOTE 5 A dimensional variable is similar to "associative arrays" in some programming languages, e.g. JavaScript.

NOTE 6 The 'type' attribute of a dimensional variable specifies the type of every value contained in that variable, and not the overall type of the variable (which is an array).

EXAMPLE 1 The volume of a 5.1 surround sound system can be defined as a vector of 6 integer values, each between 0 and 100. Each value represents the volume for a particular channel.

```
<variable id="volume" type="volumeType" dim="channelType" />
…
<xsd:schema>
  <xsd:simpleType name="volumeType">
     <xsd:restriction base="xsd:integer">
        <xsd:minInclusive> 0 </xsd:minInclusive>
        <xsd:maxInclusive> 100 </xsd:minInclusive>
     </xsd:restriction>
  </xsd:simpleType>
  <xsd:simpleType name="channelType">
     <xsd:restriction base="xsd:string">
        <xsd:enumeration value="FrontLeft" />
        <xsd:enumeration value="FrontCenter" />
        <xsd:enumeration value="FrontRight" />
        <xsd:enumeration value="RearLeft" />
        <xsd:enumeration value="RearRight" />
        <xsd:enumeration value="SubWoofer" />
     </xsd:restriction>
  </xsd:simpleType>
</xsd:schema>
```

NOTE 7 References (e.g. from atomic resources) to dimensional variables refer to the individual components of the variable. In the above example, the URI http://example.com/sound/socket#volume would refer to any individual volume component (channel). However, the URI http://example.com/sound/socket#dims(volume) would refer to the overall variable with all components, i.e. to the volume of the channels as a compound entity. See ISO/IEC 24752-5 for details.

NOTE 8 The order, in which the components of a dimensional variable are presented to the user, is based on the defined ordering of the pertinent index type(s). See ISO/IEC 24752-1 for details.

8.9 Variable dependencies

8.9.1 General

Variable dependencies express when a variable is relevant or writeable by the user, and what dependencies between the values of variables are existing.

All dependencies shall be coded as XPath expressions and evaluated at runtime. Some types of dependency may change during a session.

Variable dependencies shall be specified by embedding them within a variable element using the markup:

```
<dependency>
```

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 24752-2:2014

 © ISO/IEC 2014 – All rights reserved

```
   <relevant>expr1</relevant>
   <write>expr2</write>
</dependency>
```

The <dependency> element may be present as subelement of <variable> (see 8.1). If present it shall occur exactly once.

8.9.2 The <relevant> dependency

The <relevant> dependency specifies when a variable may be presented to the user, i.e. how relevant it is based on a particular state of the target. If the <relevant> dependency is true, the variable shall be presented to the user. This does not necessarily mean that it is directly accessible in a user interface derived from the socket, but that there is some navigation path by which the user can find it.

The <relevant> element may be present as subelement of <dependency> (see 8.9.1). If present, it shall occur exactly once. If not present, the variable inherits the <relevant> dependency from the closest ancestor <set> element that has a <relevant> dependency specified (see 7.4.2).

EXAMPLE A variable representing an enrolment number may be only available if the 'userType' variable has the value "student":

```
<dependency>
   <relevant> uis:value('userType') eq 'student' </relevant>
</dependency>
```

The content of the <relevant> element shall be a valid XPath expression (with allowed extensions as specified in 5.2) that evaluates to a Number or Boolean value. Numbers shall be in the range of [0.0, 1.0] (both 0.0 and 1.0 are included). This allows for any number of ordered degrees of relevance, with higher numbers meaning "more relevant" than lower numbers. 0.0 represents "not relevant", and 1.0 represents "relevant". If the XPath expression evaluates to a Boolean value, "true" shall be interpreted as 1.0, and "false" as 0.0.

NOTE The result of the XPath expression may change during a session.

If no <relevant> dependency is specified (neither on the variable nor on any of its ancestor sets) or if it is empty, the <relevant> dependency of the variable shall be always true, even when a <notify> element is active.

8.9.3 The <write> dependency

The <write> dependency specifies whether the user can change the value of a variable. If the <write> dependency is false, then it is inappropriate to modify the variable's value; otherwise, it is appropriate to attempt to set a new value, although there is no guarantee that it will succeed.

The <write> element may be present as subelement of <dependency> (see 8.9.1) if the 'final' attribute (see 8.7) is "false" or not present on the pertaining variable. If present, it shall occur exactly once. If not present, the variable inherits the <write> dependency from the closest ancestor <set> element that has a <relevant> dependency specified (see 8.9.3).

EXAMPLE 1 A hotel reservation system may have a variable for the type of credit payment that can only be written if the user has opted to pay by credit card. This dependency can be expressed as:

```
<variable id="creditCard" type="creditCardType">
   <dependency>
      <write> uis:value('creditPayment') </write>
   </dependency>
</variable>
```

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 24752-2:2014
Withdrawn

© ISO/IEC 2014 – All rights reserved

EXAMPLE 2 Dependencies can be used to represent ordering constraints. For example, a service's variables may only become writeable after the user has accepted the terms of a license. The user's acceptance can be represented as a Boolean variable with id="warrantySigned", and the other variables would have write attributes of the form:

```
<variable id="serviceVar" type="serviceVarType">
   <dependency>
      <write> uis:value('warrantySigned') </write>
   </dependency>
</variable>
```

The content of the <write> element shall be either empty or a valid XPath expression (with allowed extensions as specified in 5.2) that evaluates to a Boolean value.

NOTE 1 The result of the XPath expression may change during a session.

If a variable's <write> element is empty or if it inherits an empty <write> dependency from an ancestor set the <write> dependency of the variable shall be unknown.

NOTE 2 The introduction of an unknown value is analogous to a three-valued Boolean logic that allows for uncertainty. If the <write> dependency is unknown, the URC has no other choice than to find out by trial whether it can change the variable or not.

If no <write> dependency is specified (neither on the variable nor on any of its ancestor sets), the <write> dependency of the variable shall be always true, even when a <notify> element is active.

8.9.4 The <insert> dependency

The <insert> dependency specifies whether the user can modify the pertaining set of valid index combinations at runtime (with the pertaining set being the indices specified by the 'dim' attribute on the corresponding <variable> element, see 8.1).

The content of the <insert> dependency shall be an XPath expression that evaluates to a Boolean value which may change during a session. If it evaluates to false, then it is inappropriate to add or delete an index combination; otherwise it is appropriate to attempt to add or delete an index combination, although there is no guarantee that it will succeed.

NOTE 1 "Adding an index combination" means to add a variable component for a particular new index combination within the index space defined by the corresponding 'dim' attribute. Note that the initial value of the new variable component is determined by the target. "Removing an index combination" means to delete a variable component for a particular index combination that is currently occurring.

The <insert> element may be present as subelement of <dependency> (see 8.9.1) only if the corresponding variable has a 'dim' attribute (see 8.8), and is not final (see 8.7). If present, it shall occur exactly once.

Its default value shall be "false()".

NOTE 2 There is no inheritance from the <insert> dependency of a <set> element to that of a contained <variable> element (see 8.1). Each <insert> dependency pertains only to the set of indices that are specified by the 'dim' attribute of the corresponding level.

EXAMPLE With the following socket description, illustrating a table with rows and columns, the URC can request to add or remove a row (index), but not a column (index).

```
<set id="row" dim="xsd:positiveInteger">
   <dependency>
         <insert> true() </insert>
   </dependency>
   <variable id="column" type="xsd:integer" dim="xsd:positiveInteger">
         <dependency>
         <insert> false() </insert>
         </dependency>
   </variable>
```

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 24752-2:2014
Withdrawn

 © ISO/IEC 2014 – All rights reserved

```
</set>
```

8.9.5 The <length> dependency

The <length> element specifies an XPath expression (with syntax as defined in 5.2) that evaluates to a number. This number specifies the allowed length of the variable at runtime. It shall not change during a session.

NOTE The behaviour in case of a changing <length> dependency is undefined. For example, a URC may ignore any change. A conformance checking tool may reject <length> dependencies that reference socket elements other than final variables.

The <length> element may occur as subelement of <dependency> (see 8.9.1), but only if the type of the pertaining variable is xsd:string, or derived from xsd:string. If present, it shall occur once.

EXAMPLE 1 The following specifies a length constraint of 5:

```
<length> 5 </length>
```

EXAMPLE 2 The following specifies that the value of the (final) variable with id 'lengthVar' be used as length constraint:

```
<length> uis:value("lengthVar") </length>
```

The <length> element should only be used if the allowed length has to be determined at session initialization, in relation to other socket elements. If a variable has a length constraint that can be determined before runtime, it should express this constraint through a type that is defined in the <xsd:schema> section (see Clause 11) of the socket description. If a variable has a length constraint on both its type and through a <length> element, they shall result in the same number.

8.9.6 The <minLength> dependency

The <minLength> element specifies an XPath expression (with syntax as defined in 5.2) that evaluates to a number. This number specifies the minimum length of the variable at runtime. It shall not change during a session.

NOTE The behaviour in case of a changing <minLength> dependency is undefined.

The <minLength> element may occur as subelement of <dependency> (see 8.9.1), but only if the type of the pertaining variable is xsd:string, or derived from xsd:string. If present, it shall occur once.

The <minLength> element should only be used if the allowed minimal length is determined at session initialization, in relation to other socket elements. If a variable has a minimal length constraint that can be determined before runtime, it should express this constraint through a type that is defined in the <xsd:schema> section (see Clause 11) of the socket description. If a variable has a minimal length constraint on both its type and through a <length> element, the variable shall comply with both (i.e. the higher number is taken as minimal length).

8.9.7 The <maxLength> dependency

The <maxLength> element specifies an XPath expression (with syntax as defined in 5.2) that evaluates to a number. This number specifies the maximum length of the variable at runtime. It shall not change during a session.

NOTE The behaviour in case of a changing <maxLength> dependency is undefined.

The <maxLength> element may occur as subelement of <dependency> (see 8.9.1), but only if the type of the pertaining variable is xsd:string, or derived from xsd:string. If present, it shall occur once.

The <maxLength> element should only be used if the allowed maximal length is determined at session initialization, in relation to other socket elements. If a variable has a maximal length constraint that

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 24752-2:2014
Withdrawn

can be determined before runtime, it should express this constraint through a type that is defined in the <xsd:schema> section (see Clause 11) of the socket description. If a variable has a maximal length constraint on both its type and through a <length> element, the variable shall comply with both (i.e. the lower number is taken as maximal length).

8.9.8 The <pattern> dependency

The <pattern> element specifies an XPath expression (with syntax as defined in 5.2) that evaluates to a string. This string specifies a regular expression that defines a pattern for the allowed values of the variable at runtime. The pattern syntax is as specified for the 'pattern' constraining facet in XML Schema Part 2. The pattern shall not change during a session.

NOTE 1 The behaviour in case of a changing <pattern> dependency is undefined.

The <pattern> element may occur once as subelement of <dependency> (see 8.9.1), but only if the type of the pertaining variable is xsd:string, or derived from xsd:string.

NOTE 2 If a variable's value space needs to be described by a union of patterns, these can be ORed together by the "|" operator inside a regular expression.

EXAMPLE 1 The following specifies a pattern for the string-based representation of any positive integer:

```
<pattern> "[1-9][0-9]*" </pattern>
```

EXAMPLE 2 The following specifies that the value of the (final) variable 'patternVar' be used as pattern:

```
<pattern> uis:value("patternVar")</pattern>
```

NOTE 3 If any of the characters '&', '<' or '> ' occur inside <pattern> they need to be XML escaped.

EXAMPLE 3 The following specifies a pattern for strings like "A&A", "A&B", etc.

```
<pattern> "[A-Z]&amp;[A-Z]" </pattern>
```

The <pattern> element should only be used if the specified pattern is determined at session initialization, in relation to other socket elements. If a variable has a pattern constraint that can be determined before runtime, it should express this constraint through a type that is defined in the <xsd:schema> section (see Clause 11) of the socket description. If a variable has pattern constraints on both its type and through <pattern> elements, the variable shall comply with all (i.e. they are ANDed together).

8.9.9 The <minInclusive> dependency

The <minInclusive> element specifies an XPath expression (with syntax as defined in 5.2) that evaluates to a value of the same type as the variable it belongs to. This value specifies the minimum value (inclusively) of the variable at runtime. It shall not change during a session.

NOTE 1 The behaviour in case of a changing <minInclusive> dependency is undefined.

The <minInclusive> element may occur as subelement of <dependenc> (see 8.9.1), but only if the pertaining variable has a type whose value space is totally ordered. If present, it shall occur once.

NOTE 2 In XML, built-in datatypes are unordered (e.g. xsd:string) or only partially ordered (e.g. xsd:time). For these datatypes, <minInclusive> and other runtime constraints that require a total order relation cannot be used.

EXAMPLE 1 The following specifies 10 as an inclusive lower bound for a variable of type xsd:integer:

```
<minInclusive> 10 </minInclusive>
```

EXAMPLE 2 The following specifies double of the value of a final variable with id 'lowerBound' as an inclusive lower bound for a variable of type xsd:integer:

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 24752-2:2014
Withdrawn

 © ISO/IEC 2014 – All rights reserved

```
<minInclusive> 2 * uis:value("lowerBound") </minInclusive>
```

The <minInclusive> element should only be used if the allowed minimal value is determined at session initialization, in relation to other socket elements. If a variable has a minimal value constraint that can be determined before runtime, it should express this constraint through a type that is defined in the <xsd:schema> section (see Clause 11) of the socket description. If a variable has a minimal value constraint on both its type and through a <minInclusive> element, the variable shall comply with both (i.e. the higher value is taken as minimal value).

8.9.10 The <maxInclusive> dependency

The <maxInclusive> element specifies an XPath expression (with syntax as defined in 5.2) that evaluates to a value of the same type as the variable it belongs to. This number specifies the maximum value (inclusively) of the variable at runtime. It shall not change during a session.

NOTE The behaviour in case of a changing <maxInclusive> dependency is undefined.

The <maxInclusive> element may occur as subelement of <dependency> (see 8.9.1), but only if the pertaining variable has a type whose value space is totally ordered. If present, it shall occur once.

The <maxInclusive> element should only be used if the allowed maximal value is determined at session initialization, in relation to other socket elements. If a variable has a maximal value constraint that can be determined before runtime, it should express this constraint through a type that is defined in the <xsd:schema> section (see Clause 11) of the socket description. If a variable has a maximal value constraint on both its type and through a <maxInclusive> element, the variable shall comply with both (i.e. the lower value is taken as maximal value).

8.9.11 The <minExclusive> dependency

The <minExclusive> element specifies an XPath expression (with syntax as defined in 5.2) that evaluates to a value of the same type as the variable it belongs to. This number specifies the minimum value (exclusively) of the variable at runtime. It shall not change during a session.

NOTE The behaviour in case of a changing <minExclusive> dependency is undefined.

The <minExclusive> element may occur as subelement of <dependency> (see 8.9.1), but only if the pertaining variable has a type whose value space is totally ordered. If present, it shall occur once.

The <minExclusive> element should only be used if the allowed minimal value is determined at session initialization, in relation to other socket elements. If a variable has a minimal value constraint that can be determined before runtime, it should express this constraint through a type that is defined in the <xsd:schema> section (see Clause 11) of the socket description. If a variable has a minimal value constraint on both its type and through a <minExclusive> element, the variable shall comply with both (i.e. the higher value is taken as minimal value).

8.9.12 The <maxExclusive> dependency

The <maxExclusive> element specifies an XPath expression (with syntax as defined in 5.2) that evaluates to a value of the same type as the variable it belongs to. This number specifies the maximum value (exclusively) of the variable at runtime. It shall not change during a session.

NOTE The behaviour in case of a changing <maxExclusive> dependency is undefined.

The <maxExclusive> element may occur as subelement of <dependency> (see 8.9.1), but only if the pertaining variable has a type whose value space is totally ordered. If present, it shall occur once.

The <maxExclusive> element should only be used if the allowed maximal value is determined at session initialization, in relation to other socket elements. If a variable has a maximal value constraint that can be determined before runtime, it should express this constraint through a type that is defined in the <xsd:schema> section (see Clause 11) of the socket description. If a variable has a maximal value

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 24752-2:2014
WITHDRAWN

© ISO/IEC 2014 – All rights reserved

constraint on both its type and through a <maxExclusive> element, the variable shall comply with both (i.e. the lower value is taken as maximal value).

8.10 Selection

8.10.1 General

A list variable (i.e. a variable whose type has been derived by list) may provide a set of values that either restrict a variable's value space (closed selection) or provide suggested values for user input (open selection). The list items shall be of the same type as the restricted variable, or a subtype thereof. The set of values is described by a <selection> element for the restricted variable. Both static and dynamic value sets may be defined, and these may be combined within a single <selection> element.

The <selection> element may be present in a <variable> element (see 8.1). If present, it shall occur once.

EXAMPLE `<selection> … </selection>`

8.10.2 The 'closed' attribute

The 'closed' attribute of Boolean type may accompany the <selection> element to indicate whether the set of values is closed or open. The default value is "true", i.e. a closed selection.

EXAMPLE `<selection closed="false"> … </selection>`

The value "false", i.e. an open selection, indicates a selection containing optional suggested user values, i.e. a user may also enter a value not given within the set of values.

8.10.3 Static and dynamic selection sets

8.10.3.1 General

A selection shall be composed of one or more selection sets which may be either static or dynamic. Each selection set shall be described with a separate <selectionSetStatic> or <selectionSetDynamic> element, and shall have an 'id' attribute of type xsd:ID, that can be used to refer to the set.

EXAMPLE A closed selection composed of two selection sets: one with a static set of cities (myStaticCities) that is based on the definition of the type staticCityType; and another one with a dynamic set of cities (myDynamicCities) that is specified at run-time by content of the variable dynCitiesVariable. Note that staticCityType is derived from uis:stringListItem (see example in 11.3) which is derived from xsd:string (see 11.3); and dynCitiesVariable is of type uis:stringList.

```
<selection>
   <selectionSetStatic id="myStaticCities" typeRef="staticCityType"/>
   <selectionSetDynamic id="myDynamicCities" varRef="dynCitiesVariable"/>
</selection>
```

8.10.3.2 The <selectionSetStatic> element

A static selection set (i.e. that does not change at runtime) is denoted by a <selectionSetStatic> element. It shall have a 'typeRef' attribute which contains the name of a type.

The referenced type shall be derived by restriction from a simple type. Unions of enumerations are also valid, and provide a mechanism by which structure within the selection set can be captured. See 11.4 for an example of such a type.

Valid selections are all elements conforming to the given type.

NOTE User agents may render the set of values in a way that reflects the structure of the referenced types and the selection set elements, where such structure is present. For example, dividers could separate different groups of options of a visual combo-box in a GUI environment.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 24752-2:2014
Withdrawn

 © ISO/IEC 2014 – All rights reserved

8.10.3.3 The <selectionSetDynamic> element

A dynamic selection set shall be denoted by a <selectionSetDynamic> with a 'varRef' attribute which shall contain the id of a variable of type uis:csvlist, or the id of a space-separated value list variable (see 8.3.3) whose member type is compatible with the type of the host variable. The referenced list variable shall not be dimensional.

Valid selections are all elements that are currently held by the list variable.

If the host variable is not dimensional, and is not contained in one or more sets that are dimensional, the referenced list variable shall not be contained in one or more sets that are dimensional. If the host variable is dimensional or is contained in one or more sets that are dimensional, and if the referenced list variable is contained in one or more sets that are dimensional, the relevant path for the list variable shall be determined by sharing the indices of the sets that are common with the path for the variable with the 'varRef' attribute.

EXAMPLE In a television set, the variable "channel" allows only for values that are contained in the whitespace separated list given by the value of the variable "channelList". During run-time the value of the variable "channelList" may change at any time, thus changing the set of available values of the variable "channel".

```
<variable id="channel" type="xsd:string">
   <selection closed="true">
      <selectionSetDynamic id="currentChannels" varRef="channelList" />
   </selection>
</variable>
<variable id="channelList" type="uis:stringList" />
```

8.11 Platform-specific mapping information for variables

The <mapping> element may be used any number of times as subelement of <variable> (see 8.1) to include platform-specific mapping information for the variable.

A <mapping> element shall have a 'platform' attribute whose value is not restricted by ISO/IEC 24752.

A <mapping> element may have arbitrary element content and subelements. However, subelements shall be from namespaces other than the uis namespace.

EXAMPLE UPnP-specific mapping information for a variable may be included as follows:

```
<mapping platform="UPnP">
…
</mapping>
```

NOTE 1 Socket descriptions that contain platform specific mapping information lose their platform neutrality. Although multiple mappings may be specified in a socket description (one for each platform) it is recommended to consider other mechanisms of specifying the binding to platform-specific technologies. For example, mapping information may be provided in an external file with references to the elements of the socket description.

NOTE 2 Vendors and platform carriers are strongly discouraged from using the <mapping> element for embedding active or executable content in a socket description. This would introduce a security risk for components parsing such a socket description, and executing such content.

8.12 Variable properties from DCMI

Any element and element refinement from ISO 15836, Dublin Core Metadata Element Set, or the set of Dublin Core Metadata Initiative (DCMI) Metadata Terms may be used to describe a socket variable, if appropriate. Each of them may occur multiple times as subelement of <variable> (see 8.1).

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 24752-2:2014
Withdrawn

© ISO/IEC 2014 – All rights reserved

9 Commands

9.1 General

Command elements are used to capture commands that a user may issue to a target. A command is a core function that a user can request a target to perform and that cannot be represented by a variable. A command should represent some function beyond simply manipulating the value of a single variable.

EXAMPLE The seek button on a CD player and a submit button on an online form.

The <command> element may occur as subelement of <set> (see 7.1) or as subelement of <uiSocket> (see 6.1).

9.2 The 'id' attribute

A <command> element (see 9.1) shall have an 'id' attribute.

The 'id' attribute shall be of type ID as defined by XML Schema Part 2: Datatypes. It provides an identifier which is used to refer to the element within the socket description and as a means of binding externally defined resources such as labels. The 'id' value shall be unique among all 'id' and 'name' attributes within the socket description.

EXAMPLE `<command id="seek">`

NOTE 'id' attribute values are not normally presented to users and need not be human comprehensible.

9.3 The 'type' attribute

9.3.1 General

<command> elements (see 9.1) may have a 'type' attribute. The type of a command describes the states that the command may have. The state provides information about the last execution (request) of the command.

The 'type' attribute takes a QName as value (see XML Schema Part 2: Datatypes). If the namespace prefix is not specified, the namespace http://openurc.org/ns/uisocketdesc-2 is assumed.

EXAMPLE 1 The following type values are semantically the same: type="uis:voidCommand" and type="voidCommand", assuming that the namespace identifier uiSocket is bound to http://openurc.org/ns/uisocketdesc-2.

The following subsections define three command types. The default command type shall be uis:voidCommand.

9.3.2 uis:voidCommand

type="uis:voidCommand": the command shall be stateless.

A command of type uis:voidCommand shall not have output or input-output parameters.

NOTE This restriction is necessary since a command with output or input-output parameters cannot execute multiple instances of the same command at one time, since there is only one set of output parameters that can receive the results.

9.3.3 uis:basicCommand

type="uis:basicCommand": the command shall have the following set of states available: "initial", "rejected", "inProgress", "done", "succeeded", "failed".

The meanings of the states shall be as follows:

— "initial": The command has not been requested for execution

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 24752-2:2014
Withdrawn

 © ISO/IEC 2014 – All rights reserved

— “rejected”: The command’s execution has been requested, but the target rejected the request

— “inProgress”: The command is currently being executed as requested

— “done”: The command has been executed

— “succeeded”: The command has been executed with success (this is more specific than “done” and should be preferred if known by the target)

— “failed”: The command was executed but failed (this is more specific than “done” and should be preferred if known by the target)

NOTE 1 Upon failure of a command’s execution the target would typically convey an error message to the user in a separate way, e.g. by raising a notification.

NOTE 2 The state of a command says nothing about whether this command may be executed (again). This is specified by the <write> dependency (see 9.9.3).

NOTE 3 A command of type “uis:basicCommand” cannot be invoked multiple times on the target unless the previous invocation has completed (see ISO/IEC 24752-1).

EXAMPLE A pressed floor button in an elevator is a command with state “inProgress”.

9.3.4 uis:timedCommand

type=”uis:timedCommand”: the same as uis:basicCommand except that the state “inProgress” shall make an additional “ttc” field available.

The “ttc” field shall be of type xsd:duration.

The value of the “ttc” field may be undefined at any time.

If the command is in state “inProgress”, the target may periodically “count down” the value of the “ttc” field until the command switches to another state. The value of the “ttc” field (if not the empty string) shall be a hint from the target to the URC as to how long the estimated time to complete is – no guarantee is provided by the target whatsoever. The “ttc” field shall be read-only for the URC. The value of the “ttc” field shall be invalid in any other state than “inProgress”.

NOTE 1 A command of type uis:timedCommand cannot be invoked multiple times on the target unless the previous invocation has completed (see ISO/IEC 24752-1).

NOTE 2 The value of the “ttc” field can be accessed in dependency expressions by appending “[ttc]” to the path in the XPath function uis:value(path), see 5.2.5.2.

EXAMPLE A “book” command in an online ticket system is of type uis:timedCommand and conveys the estimated time for the booking process in real-time.

9.4 The ‘sensitive’ attribute

A <command> element (see 9.1) may have a ‘sensitive’ attribute. It shall be a Boolean indicating whether the command represents a legally sensitive action.

The default value shall be “false”.

EXAMPLE An emergency call button command whose form of presentation has legal implications would be described with the markup:

```
<command id=”emergency” sensitive=”true” />
```

9.5 The ‘sufficient’ attribute

A <command> element (see 9.1) may have a ‘sufficient’ attribute of type Boolean.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 24752-2:2014
Withdrawn

© ISO/IEC 2014 – All rights reserved

The value "true" indicates that the command is sufficiently specified. A command is sufficiently specified iff its set of assert dependencies, if any, is sufficient. A set of assert dependencies is sufficient iff whenever all their expressions evaluate to true, the command has completed successfully on the target. The 'assert' dependency is specified in 9.9.5.

NOTE 1 The spelling "iff" above and below should be read "if and only if" and means that the statements before and after "iff" are equivalent. So if any one of them is true, the other is true, too. (Also, if any one of them is false, the other is false, too.)

EXAMPLE For the "floor 4" command button of an elevator, the following postcondition is not sufficient, because the command could fail and the value of the expression could still be true:

```
<dependency>
   <assert> uis:value('currentFloor') > 0</assert>
</dependency>
```

However, the following postcondition is sufficient:

```
<dependency>
   <assert> uis:value('currentFloor') eq 4 </assert>
</dependency>
```

NOTE 2 In the case of sufficient descriptions of commands, an intelligent URC can apply advanced inference techniques and thus provide a more usable user interface. For example, it can conclude that a command doesn't need to be invoked if its assert dependencies are already true.

If present, the 'sufficient' attribute of <command> (see 9.1) overrides the 'sufficient' attribute of <uiSocket> (see 6.4), but only for that particular command. If not present, the value of the 'sufficient' attribute of <uiSocket> applies to the command.

9.6 The 'complete' attribute

The <command> element (see 9.1) may have a 'complete' attribute of type Boolean.

The value "true" indicates that all commands in the socket description are completely specified with regard to their parameters. A command is completely specified iff its input and output parameter sets are complete. Command parameters are specified in 9.12.

A set of command input parameters is complete iff the outcome of a successful execution of the command is guaranteed to be the same whenever the command is executed with the same set of input parameter values.

A set of command output parameters is complete iff no socket variables other than those specified as output parameters are modified as a result of the command's execution for all possible sets of input parameters.

NOTE 1 The spelling "iff" above and below should be read "if and only if" and means that the statements before and after "iff" are equivalent. So if any one of them is true, the other is true, too. (Also, if any one of them is false, the other is false, too.)

NOTE 2 In the case of complete descriptions of commands, an intelligent URC can apply advanced inference techniques and thus provide a more usable user interface.

If present, the 'complete' attribute of <command> (see 9.1) overrides the 'complete' attribute of <uiSocket> (see 6.5), but only for that particular command. If not present, the value of the 'complete' attribute of <uiSocket> applies to the command.

9.7 The 'optional' attribute

The <command> element (see 9.1) may have an 'optional' attribute of type Boolean. Its default value shall be "false".

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 24752-2:2014
Withdrawn

 © ISO/IEC 2014 – All rights reserved

A value of “true” indicates that the command may not be available at runtime due to various constraints.

NOTE 1 Examples for constraints that impact the availability of a socket element are: access control, different products using a common socket description but with some implementation variation (such as in UPnP DCPs).

NOTE 2 The availability of a socket element will be indicated at runtime to the URC through TUN-specific means (see ISO/IEC 24752-1).

9.8 The ‘dim’ attribute

The <command> element (see 9.1) may have a ‘dim’ attribute specifying a command as dimensional (with one or more dimensions). At runtime a dimensional command shall have multiple states, each one for a particular combination of indices.

If present, the ‘dim’ attribute shall contain a non-empty, ordered, space-separated list of type references that are the command’s index types. The first reference specifies the index type for the first dimension, the second type for the second dimension, and so on. Valid index type references are: (a) the name of a type that is defined (see 11.5) or imported (see 11.6) in the <xsd:schema> part of the socket description, and (b) the fully qualified name (QName) of an external type (see 11.7).

A command’s index shall not have the value “ttc” which is reserved to denote its “ttc” field (see 5.2.5.2).

NOTE 1 The ‘type’ attribute of a dimensional command specifies the type of every single value contained in that command, and not the overall type of the command.

NOTE 2 There is no explicit ordering defined for the states resulting from a dimensional command. However, ordering the states based on the order of the index values is recommended for the target — in conveying the states to the URC — and for the URC — in presenting the states to the user (see ISO/IEC 24752-1).

NOTE 3 The largest possible set of values for a particular socket command results from the product of all index types that occur when walking the path from the <uiSocket> element down to the particular socket command (see definition of path in 5.2.5.2). However, not all index combinations may actually occur at runtime.

NOTE 4 The index type may specify an infinite number of possible index values (e.g. xsd:integer has infinite values). However, the actual index values at runtime are a finite subset of the values allowed by the index type.

NOTE 5 The use of a dimensional command is only reasonable if the set of commands is really homogenous, i.e. each command has identical characteristics including label (message) and help text.

NOTE 6 References (e.g. from atomic resources) to dimensional commands refer to the individual components of the command. For example, the URI http://example.com/thermometer/socket#reset would refer to any single command component within the dimensional command with id=’reset’. However, the URI http://example.com/thermometer/socket#dims(reset) would refer to the overall command with all components. See ISO/IEC 24752-5 for details.

NOTE 7 The order, in which the components of a dimensional command are presented to the user, is based on the defined ordering of the pertinent index type(s). See ISO/IEC 24752-1 for details.

9.9 Command dependencies

9.9.1 General

Command dependencies express when a command is relevant or executable by the user, and what a command’s effect is (“postcondition”).

Command dependencies shall be specified by embedding them within a command element using the markup:

```
<dependency>
   <relevant>expr1</relevant>
   <write>expr2</write>
</dependency>
```

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 24752-2:2014

The <dependency> element may occur once as subelement of <command> (see 9.1).

9.9.2 The <relevant> dependency

The <relevant> dependency specifies when a command may be presented to a user, i.e. how relevant it is based on a particular state of the target. If the <relevant> dependency is true, the command shall be presented to the user. This does not necessarily mean that it is directly accessible in a user interface derived from the socket, but that there is some navigation path by which the user can find it.

The <relevant> element may be present as subelement of <dependency> (see 9.9.1). If present, it shall occur exactly once. If not present, the command inherits the <relevant> dependency from the closest ancestor <set> element that has a <relevant> dependency specified (see 7.4.2).

EXAMPLE A command to make a travel reservation may only be relevant when the user is currently defining a reservation. This dependency can be expressed using the <relevant> dependency as follows:

```
<command id="makeReservation" type="uis:basicCommand">
            <dependency>
      <relevant> uis:value('activity') eq 'makingReservation' </relevant>
   </dependency>
</command>
```

The content of the <relevant> element shall be as specified for the <relevant> dependency of variables (see 8.9.2).

If no <relevant> dependency is specified (neither on the command nor on any of its ancestor sets) or if it is empty, the <relevant> dependency of the command shall be always true, even when a <notify> element is active.

9.9.3 The <write> dependency

This dependency for commands specifies whether the user can activate the command. This is called "<write> dependency" or "precondition". If the precondition is false, then it is inappropriate to perform the action; otherwise it is appropriate to attempt to perform the action, although there is no guarantee that it will succeed.

The <write> element may be present as subelement of <dependency> (see 9.9.1). If present, it shall occur exactly once. If not present, the command shall inherit the <write> dependency from the closest ancestor <set> element that has a <write> dependency specified (see 8.9.3).

EXAMPLE An elevator button commanding the elevator to go to level 4 only be triggered if the elevator is not already at level 4. This dependency can be expressed as:

```
<dependency>
   <write> uis:value('currentFloor') ne 4 </write>
</dependency>
```

The content of the <write> element shall be as specified for the <write> dependency of variables (see 8.9.3).

If a command's <write> dependency is empty or if it inherits an empty <write> dependency from an ancestor set the <write> dependency of the command shall be unknown.

NOTE The introduction of an unknown value is analogous to a three-valued Boolean logic that allows for uncertainty. If the <write> dependency is unknown, the URC has no other choice than to find out by trial whether it can execute the command or not.

If no <write> dependency is specified (neither on the command nor on any of its ancestor sets), the <write> dependency of the command shall be always true, even when a <notify> element is active.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 24752-2:2014
Withdrawn

 © ISO/IEC 2014 – All rights reserved

9.9.4 The <insert> dependency

The <insert> dependency specifies whether the user can modify the pertaining set of valid index combinations at runtime (with the pertaining set being the indices specified by the 'dim' attribute on the corresponding <command> element, see 9.1).

The content of the <insert> dependency shall be an XPath expression that evaluates to a Boolean value which may change during a session. If it evaluates to false, then it is inappropriate to add or delete an index combination; otherwise it is appropriate to attempt to add or delete an index combination, although there is no guarantee that it will succeed.

NOTE 1 "Adding an index combination" means to add a command component for a particular new index combination within the index space defined by the corresponding 'dim' attribute. Note that the status of an added command component is determined by the target. "Removing an index combination" means to delete a command component for a particular index combination that is currently occurring.

The <insert> element may be present as subelement of <dependency> (see 9.9.1) only if the corresponding command has a 'dim' attribute (see 9.8). If present, it shall occur exactly once.

Its default value shall be "false()".

NOTE 2 There is no inheritance from the <insert> dependency of a <set> element (7.4.4) to that of a contained <command> element. Each <insert> dependency pertains only to the set of indices that are specified by the 'dim' attribute of the corresponding level.

9.9.5 The <assert> dependency

The <assert> dependency specifies a Boolean expression that is guaranteed to evaluate to true after the command has been successfully executed. This is called "assertion" or "postcondition".

NOTE 1 If the <assert> dependency is guaranteed to be sufficient, one can infer from a true postcondition that the command has succeeded. See 9.5 for details on the 'sufficient' attribute.

For commands of type uis:voidCommand (see 9.3.2) the postcondition is guaranteed to be true for any requested invocation of the command. For commands of type uis:basicCommand (see 9.3.3) or uis:timedCommand (see 9.3.4), the postcondition is guaranteed to be true for all invocations of the command that result in the state "succeeded".

The <assert> element may be present as subelement of <dependency> (see 9.9.1). If present, it shall occur exactly once.

EXAMPLE 1 After the successful execution of the "floor 4" button of the elevator the value of the variable 'currentFloor' will be 4. This assertion can be expressed as:

```
<dependency>
   <assert> uis:value('currentFloor') eq 4 </assert>
</dependency>
```

EXAMPLE 2 This "startPlay" command will trigger the target requesting the URC for opening a sub-session ("spawn forward request", see ISO/IEC 24752-1) on the "play" socket.

```
<command id="startPlay" type="uis:voidCommand">
   <dependency>
      <assert> uis:sessionForward("spawn", "http://example.com/target/play") </assert>
   </dependency>
</command>
```

NOTE 2 Advanced URCs may exploit knowledge on postconditions for generating more usable user interfaces. Postconditions are especially useful if they are sufficiently specified (see 9.5).

The content of the <assert> element shall be either empty or a valid XPath expression that evaluates to a Boolean value.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 24752-2:2014

© ISO/IEC 2014 – All rights reserved

If a command's <assert> dependency is empty the <assert> dependency of the command shall be unknown. If no <assert> dependency is specified, the <assert> dependency for the command shall be always true.

9.10 Platform-specific mapping information for commands

The <mapping> element may be used any number of times as subelement of <command> (see 9.1) to include platform-specific mapping information for the containing command.

A <mapping> element shall have a 'platform' attribute whose value is not restricted by the ISO/IEC 24752 series.

A <mapping> element may have arbitrary element content and subelements. However, subelements shall be from namespaces other than the uis namespace.

NOTE 1 Socket descriptions that contain platform specific mapping information lose their platform neutrality. Although multiple mappings may be specified in a socket description (one for each platform) it is recommended to consider other mechanisms of specifying the binding to platform-specific technologies. For example, mapping information may be provided in an external file with references to the elements of the socket description.

NOTE 2 Vendors and platform carriers are strongly discouraged from using the <mapping> element for embedding active or executable content in a socket description. This would introduce a security risk for components parsing such a socket description, and executing such content.

9.11 Command properties from DCMI

Any element and element refinement from ISO 15836, Dublin Core Metadata Element Set, or the set of Dublin Core Metadata Initiative (DCMI) Metadata Terms may be used to describe a socket command, if appropriate. Each of them may occur multiple times within a <command> element (see 9.1).

9.12 Command parameters

9.12.1 General

A <command> element (see 9.1) may have any number of <param> subelements, each defining one distinct parameter pertaining to the command.

NOTE 1 A command parameter's scope is either local to the command or global (in the socket). See 9.12.2 and 9.12.3 below.

EXAMPLE A command to call an elevator has three input parameters: the origin floor, the destination floor, and the wait time for the elevator door to stay open while nobody passes the door. The parameters "origin floor" and "destination floor" are local, i.e. they have to be supplied for every call. The wait time is a global parameter, i.e. it can be set as a global setting for all subsequent calls.

```
<command id="callElevator" type="uis:basicCommand">
   <param id="originFloor" dir="in" type="xsd:integer" />
   <param id="destinationFloor" dir="in" type="xsd:integer" />
   <param idref="waitTime" dir="in" />
</command>
```

NOTE 2 Advanced URCs may exploit knowledge on command parameters for generating more usable user interfaces. This information is especially useful if the command is guaranteed to be completely specified. See 9.6 for details on the 'complete' attribute.

9.12.2 The 'id' attribute (local parameter)

A <param> element (see 9.12.1) shall have either one 'id' attribute or one 'idref' attribute (see 9.12.3). A <param> element with an 'id' attribute is called "local parameter".

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 24752-2:2014
Withdrawn

 © ISO/IEC 2014 – All rights reserved

The value of local parameters shall be available to the user (i.e. has a defined state) only when the command's <relevant> dependency evaluates to true.

The value of the 'id' attribute shall be unique among all 'id' attributes within the socket description.

The 'id' attribute shall be of type ID as defined by XML Schema Part 2: Datatypes. It provides an identifier which is used to refer to the element within the socket description and as a means of binding externally defined resources such as labels. The 'id' attribute is not presented to users and need not be human comprehensible.

NOTE On command invocation, the URC will send the values of all local input parameters (i.e. local parameters with a dir value of "in" or "inout") to the target. After execution, the target will return the values of all local output parameters (i.e. local parameters with a dir value of "out" or "inout") to the URC.

9.12.3 The 'idref' attribute (global parameter)

A <param> element (see 9.12.1) shall have either one 'idref' attribute or one 'id' attribute (see 9.12.2). A <param> element with an 'idref' attribute is called "global parameter".

The 'idref' attribute shall reference a variable or notification of the same socket, i.e. the value of the 'idref' attribute shall be the value of the 'id' attribute of a <variable> or <notify> element (see 8.1) in the relevant socket description. As an exception, the 'idref' attribute may also reference a set of the same socket by its 'id', meaning that all variables contained in the set are used as global parameters for the command.

NOTE 1 Referencing a variable as global "out" parameter means that the execution of the command will impact the value of the variable. Referencing a notification as global "out" parameter means that the execution of the command will impact the status of the notification.

Local parameters of other commands shall not be referenced by an 'idref' attribute.

The 'idref' attribute is of type IDREF as defined by XML Schema Part 2: Datatypes. It specifies a constraint between the command it is contained in and a variable: If the value of the 'dir' attribute (see 9.12.4) is "in", the target will use the current value of the referenced variable as input value for the execution of the command. If the value of the 'dir' attribute is "out", the target will, after execution of the command, update the referenced value to reflect a result. If the value of the 'dir' attribute is "inout", the target will read from the referenced variable before execution and write to it after execution of the command.

The 'idref' attribute is not presented to users and need not be human comprehensible.

A global parameter shall not have a 'type' attribute (see 9.12.4.4). Instead, its type is specified by the referenced socket variable declaration (see 8.3).

NOTE 2 On command invocation, the URC will not send or receive the values of the command's global parameters. Instead, their values will be synchronized between the URC and the target as required for socket variables (as described in ISO/IEC 24752-1), independently from the command invocation.

NOTE 3 Advanced URCs may exploit the knowledge on global parameters to infer dependencies between socket elements and thus build a more usable user interface.

9.12.4 The 'dir' attribute

9.12.4.1 General

A <param> element (see 9.12.1) shall have a 'dir' attribute.

Its value shall be either "in", "out", or "inout". A value of "in" marks the parameter as input parameter. A value of "out" marks the parameter as output parameter. A value of "inout" specifies that the parameter is both input and output parameter, i.e. the parameter will be read before execution and updated after execution of the command.

NOTE Commands of type uis:voidCommand cannot have output or input-output parameters (see 9.3.2).

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 24752-2:2014
Withdrawn

© ISO/IEC 2014 – All rights reserved

9.12.4.2 Input parameters

A 'dir' attribute (see 9.12.4.1 value of "in" denotes input parameters, i.e. parameters whose value will be read by the target before execution of a command, to affect the execution and its result(s).

A command may have any number of (local and global) input parameters.

The value of a local input parameter may be changed by the URC or its user when the <write> dependency of the command evaluates to "true" or is unknown.

For global input parameters their <relevant> and <write> dependencies (see 8.9.2 and 8.9.3), specify whether they can be accessed or changed by the URC or its user.

NOTE Local input parameters are similar to socket elements, but their values are synchronized with the target only when the command is invoked, and only from the URC to the target.

9.12.4.3 Output parameters

A 'dir' attribute (see 9.12.4.1 value of "out" denotes output parameters, i.e. parameters whose value is updated by the target after execution of a command, to reflect a result of the execution.

A command may have any number of (local and global) output parameters.

The value of a local output parameter shall not be changed by the URC or its user.

For global output parameters their <relevant> and <write> dependencies (see 8.9.2 and 8.9.3), specify whether they can be accessed or changed by the URC or its user.

NOTE Local output parameters are similar to socket elements, but their values are synchronized with the target only after the command's execution has completed, and only from the target to the URC.

9.12.4.4 Input-output parameters

A 'dir' attribute (see 9.12.4.1) value of "inout" denotes input-output parameters, i.e. variables that are used as input and output parameters for the same command.

A command may have any number of (local and global) output parameters.

The value of a local input-output parameter may be changed by the URC or its user when the <write> dependency of the command evaluates to "true" or is unknown.

For global input-output parameters their <relevant> and <write> dependencies (see 8.9.2 and 8.9.3), specify whether they can be accessed or changed by the URC or its user.

NOTE 1 Local input-output parameters are similar to socket elements, but their values are synchronized with the target only in the following two cases: (1) when the command is invoked, from the URC to the target; and (2) after the command's execution has completed, from the target to the URC.

9.12.5 The 'type' attribute

A local parameter (9.12.2) shall have a 'type' attribute. A global parameter (see 9.12.3) shall not have a 'type' attribute.

The 'type' attribute value denotes the type of the parameter. Valid types are the same as for the <variable> element (see 8.3).

9.12.6 The 'secret' attribute

A local parameter (9.12.2) may have a 'secret' attribute. Its default value shall be "false".

A global parameter (see 9.12.3) shall not have a 'secret' attribute.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 24752-2:2014
Withdrawn

 © ISO/IEC 2014 – All rights reserved

The 'secret' attribute shall have the same meaning as for the <variable> element (see 8.4).

9.12.7 The 'sensitive' attribute

A local parameter (9.12.2) may have a 'sensitive' attribute. Its default value shall be "false".

A global parameter (see 9.12.3) shall not have a 'sensitive' attribute.

The 'sensitive' attribute shall have the same meaning as for the <variable> element (see 8.5).

9.12.8 The <selection> subelement

The <selection> element may be present as subelement of <param> (see 9.12.1), but only for local input (see 9.12.4.2) and local input-output parameters (see 9.12.4.4). If present, it shall occur once.

The <selection> element provides a set of values that either restricts the parameter's value space (closed selection) or provides suggested values for user input (open selection). It shall follow the same syntax as defined for variables (see 8.10).

EXAMPLE A media rendering device lets the user pick from a set of preset configurations. When invoking the command "selectPreset", the user can pick one of the presets given in the list variable "presetNameList".

```
<variable id="presetNameList" use="required" type="uis:csvlist">
   <dc:description>This variable contains a comma-separated list of valid preset names
currently supported by this device. Its value changes if/when the device changes the set
of presets that it supports. This may occur in conjunction with a vendor-defined action or
some other non-UPnP event. This state variable will include any of the predefined presets
that are supported by the device.</dc:description>
</variable>
<command id="selectPreset" use="required" type="uis:basicCommand">
   <dc:description>Set a preset configuration.</dc:description>
   <param id="presetName" type="xsd:string" dir="in">
      <selection closed="true">
         <selectionSetDynamic id="availablePresets" varRef="presetNameList"/>
      </selection>
   </param>
</command>
```

9.12.9 Platform-specific mapping information for command parameters

The <mapping> element may be used any number of times as subelement of <param> (see 9.12.1) for local parameters (see 9.12.2) to include platform-specific mapping information. It shall not occur for global parameters (see 9.12.3).

A <mapping> element shall have a 'platform' attribute whose value is not restricted by the ISO/IEC 24752 series.

A <mapping> element may have arbitrary element content and subelements. However, subelements shall be from namespaces other than the uis namespace.

NOTE 1 Socket descriptions that contain platform specific mapping information lose their platform neutrality. Although multiple mappings may be specified in a socket description, (one for each platform) it is recommended to consider other mechanisms of specifying the binding to platform-specific technologies. For example, mapping information may be provided in an external file with references to the elements of the socket description.

NOTE 2 Vendors and platform carriers are strongly discouraged from using the <mapping> element for embedding active or executable content in a socket description. This would introduce a security risk for components parsing such a socket description, and executing such content.

9.12.10 Command parameter properties from DCMI

Any element and element refinement from ISO 15836, Dublin Core Metadata Element Set, or the set of Dublin Core Metadata Initiative (DCMI) Metadata Terms may be used to describe a command parameter, if appropriate. Each of them may occur multiple times within a <param> element (see 9.12.1).

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 24752-2:2014

© ISO/IEC 2014 – All rights reserved

10 Notifications

10.1 General

Notifications are special states where normal operation is suspended, such as exception states. A target may invoke (activate) or dismiss (deactivate) a notification at any time.

EXAMPLE Examples of notifications include an announcement made by a public address system in an airport, a clock alarm, or a response to invalid input for a field of a form.

The <notify> element may occur one or more times as subelement of <set> (see 7.1) or as subelement of <uiSocket> (see 6.1).

A <notify> element shall have one of three states: “active”, “inactive”, or “stacked”. The default state is “inactive”. A target may activate or deactivate the <notify> element at any time. Some variables and commands may be only readable and writable when a notification is active, as expressed in their <relevant> and <write> dependencies (see 8.9.2, 8.9.3, 9.9.2 and 9.9.3).

A target may activate a notification while another notification is already active. In this case, the second notification takes priority over the first. The user agent shall keep a stack of notifications with the most recent on top. All active notifications that are not on the top are deemed to be in state “stacked” (this is internal to the user agent, and not shared with the target).

User interfaces based on a user interface socket shall respect the order in which notifications become active, so that the most recently activated notification is always the one presented to the user. When the top notification becomes inactive, the next notification on the stack becomes active again and is presented to the user.

Most notifications require acknowledgement by the user (depending on the ‘type’ attribute, see 10.3).

NOTE The <notify> element does not come with a text message to display to the user. Rather, the message to be displayed is given by atomic resources (in text, image or video form), as defined in resource sheets (see ISO/IEC 24752-5).

10.2 The ‘id’ attribute

A <notify> element (see 10.1) shall have an ‘id’ attribute.

The ‘id’ attribute shall be of type ID as defined by XML Schema Part 2: Datatypes. It provides an identifier which is used to refer to the element within the socket description and as a means of binding externally defined resources such as labels. The ‘id’ value shall be unique among all ‘id’ and ‘name’ attributes within the socket description.

NOTE ‘id’ attribute values are not normally presented to users and need not be human comprehensible.

EXAMPLE <notify id=”fireAlarmWarning”>

10.3 The ‘type’ attribute

A <notify> element (see 10.1) may have a ‘type’ attribute, specifying a pre-defined dialogue type.

NOTE 1 The ‘type’ attribute is a convenient mechanism for a socket developer to use built-in standard dialogue forms where suitable. The pre-defined types should be sufficient for many use cases.

NOTE 2 As for all notifications, labels, and other language-dependent user interface components for notifications using a pre-defined dialogue type can be specified as atomic resources in resource sheets (see ISO/IEC 24752-5).

If present, the ‘type’ attribute value shall have one of the following values:

— “show”: A notification shall be presented to the user. The user is not required to acknowledge the receipt of the notification (i.e. no modal dialogue is needed);

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 24752-2:2014
Withdrawn

 © ISO/IEC 2014 – All rights reserved